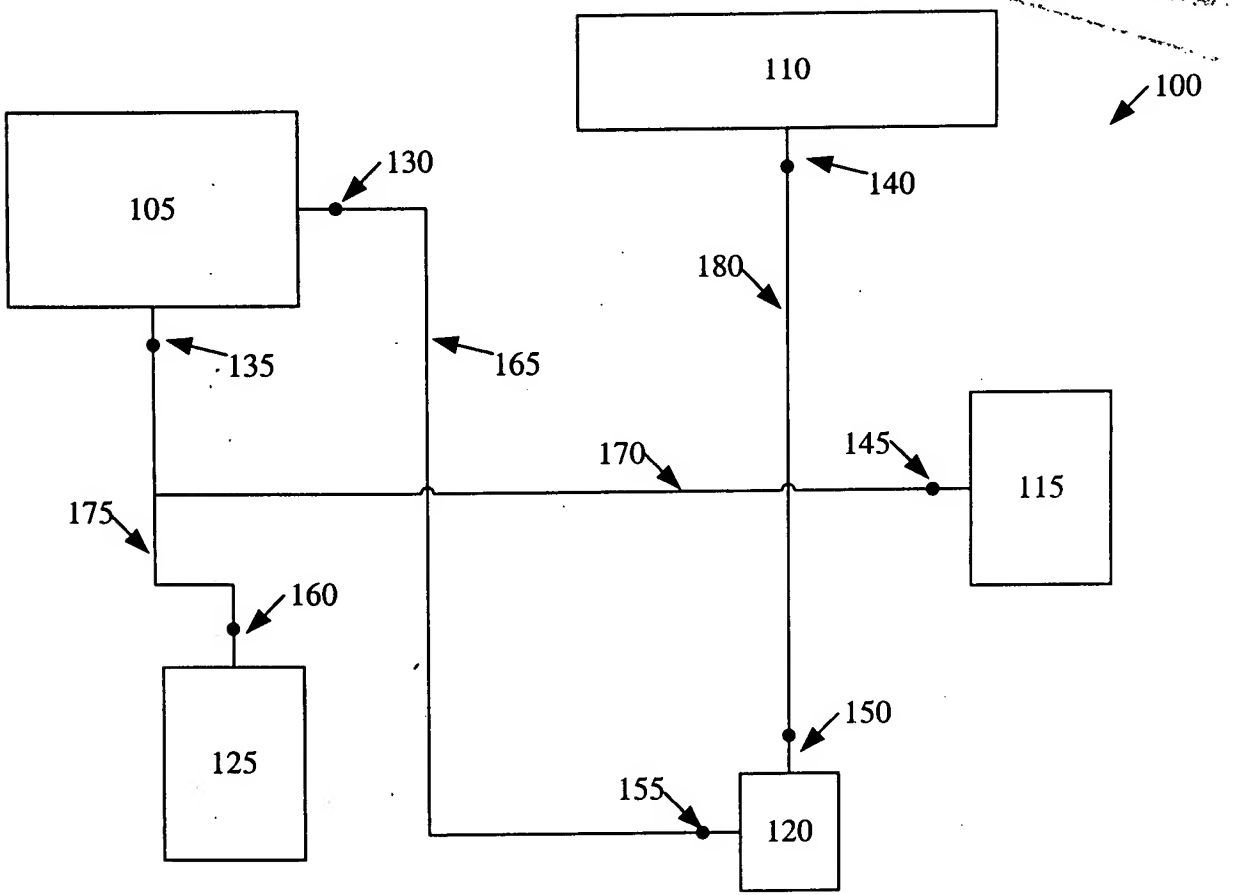
110
100
105
130
140
180
135
165
170
145
115
175
160
125
150
155
120

Figure 1

Figure 2

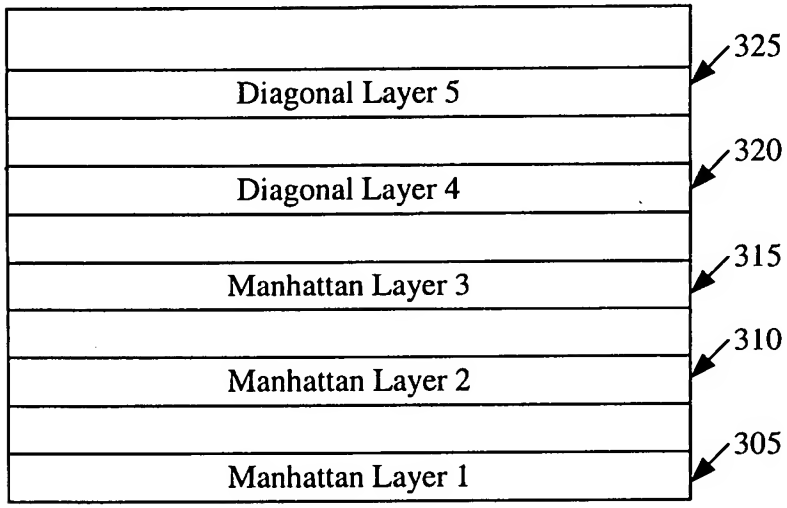
Diagonal Layer 5
Diagonal Layer 4
Manhattan Layer 3
Manhattan Layer 2
Manhattan Layer 1
325
320
315
310
305

Figure 3

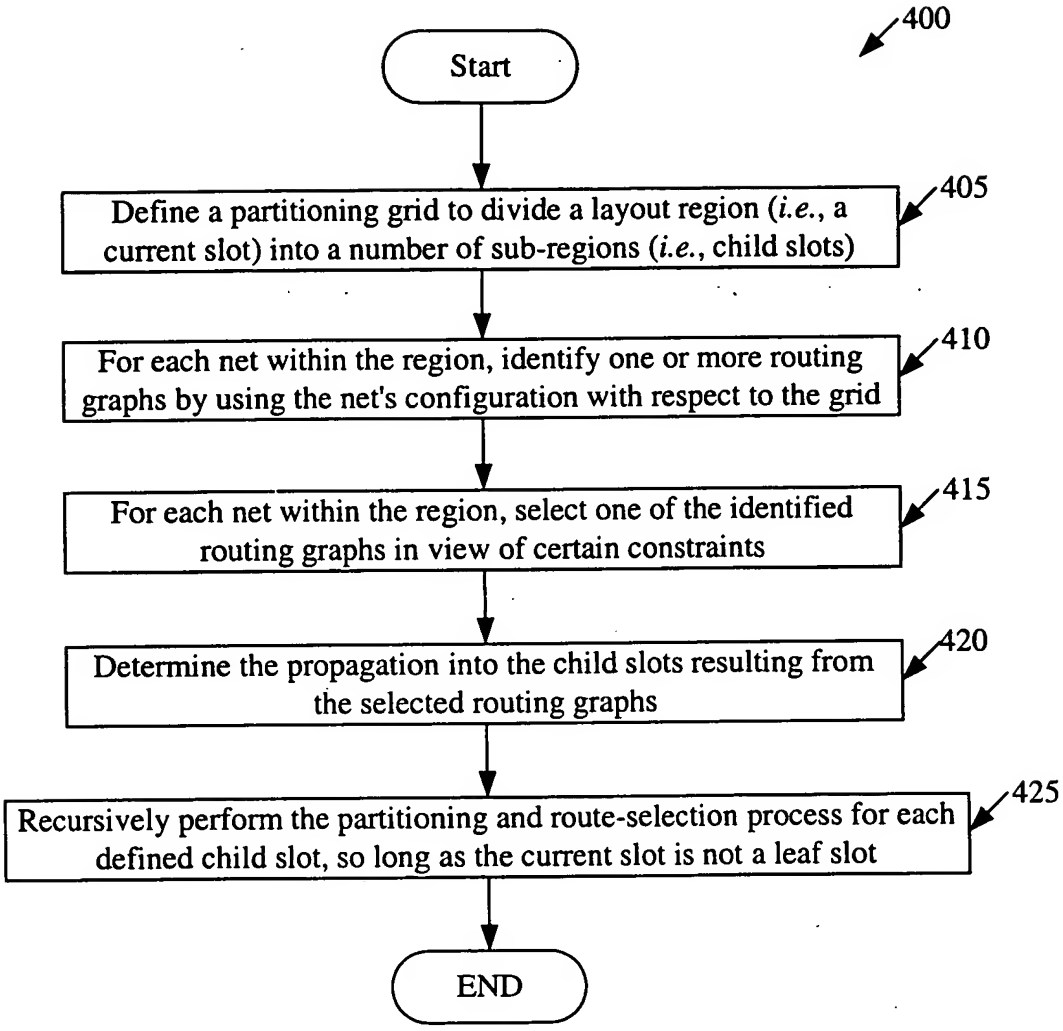
400
Start
Define a partitioning grid to divide a layout region (i.e., a current slot) into a number of sub-regions (i.e., child slots)
405
For each net within the region, identify one or more routing graphs by using the net's configuration with respect to the grid
410
For each net within the region, select one of the identified routing graphs in view of certain constraints
415
Determine the propagation into the child slots resulting from the selected routing graphs
420
Recursively perform the partitioning and route-selection process for each defined child slot, so long as the current slot is not a leaf slot
425
END

Figure 4

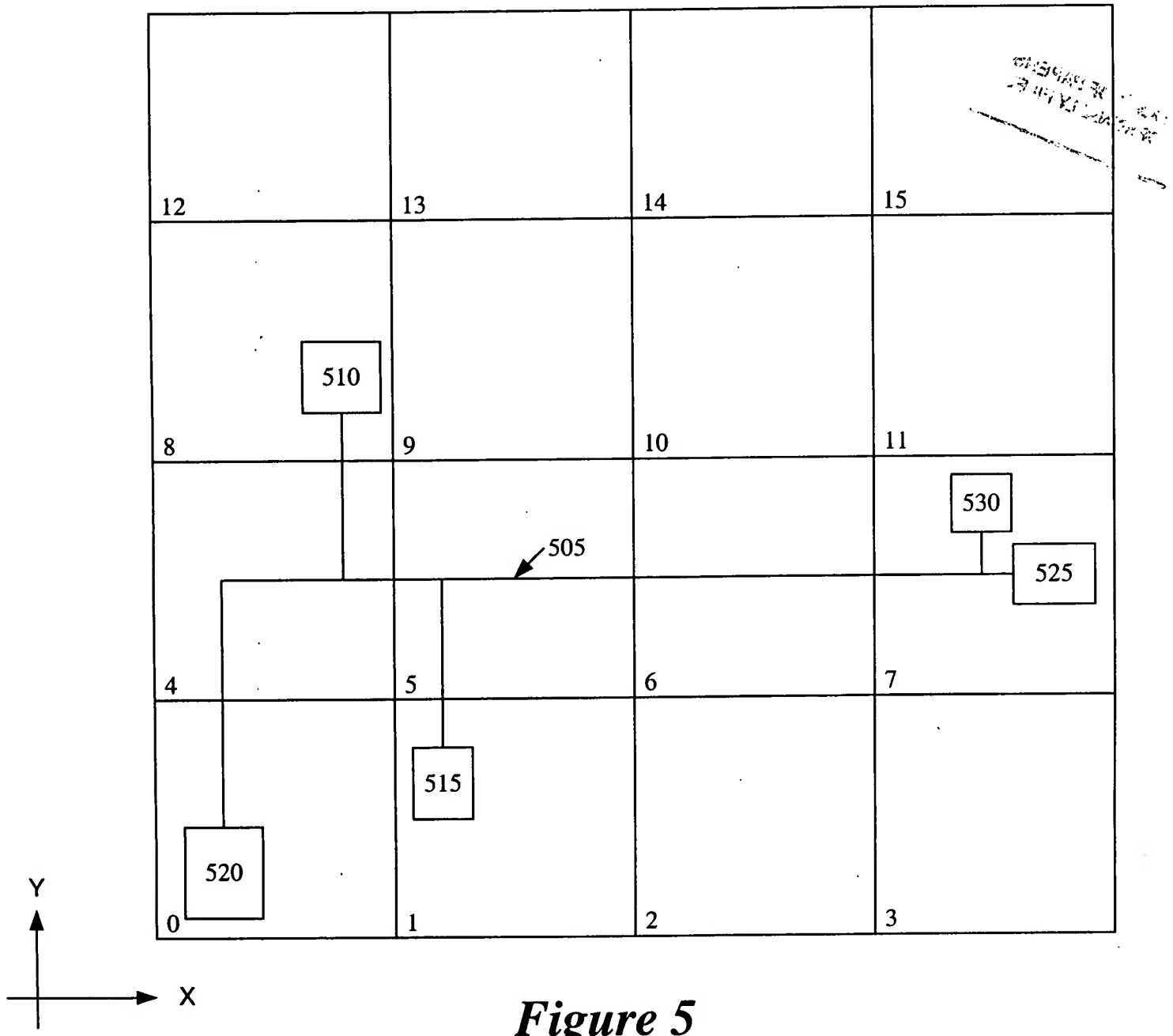

Figure 5

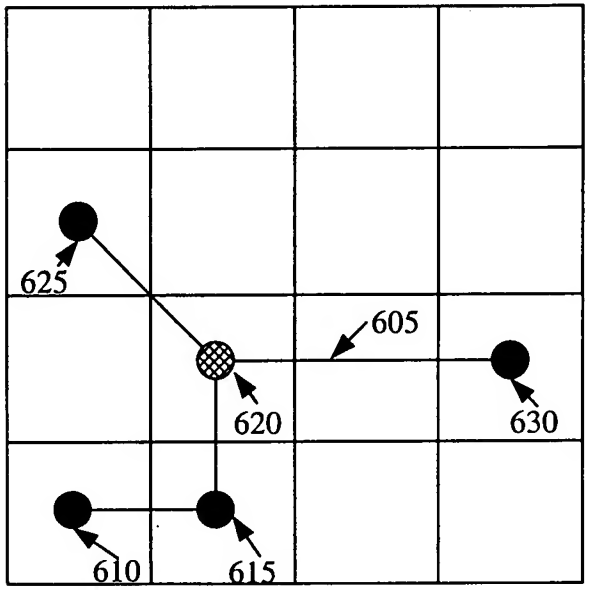

Figure 6

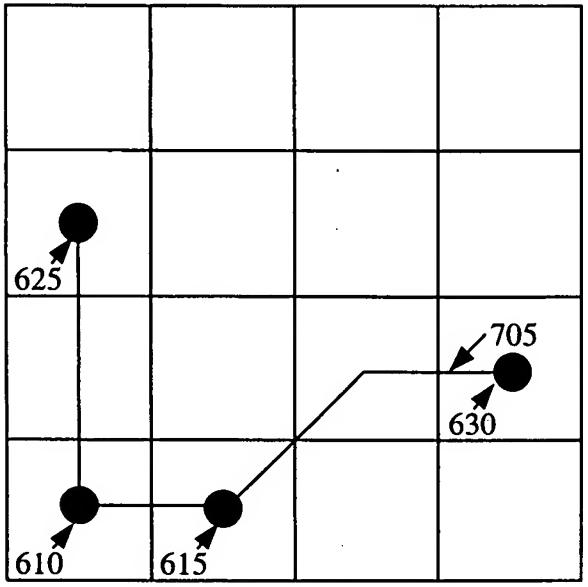

Figure 7

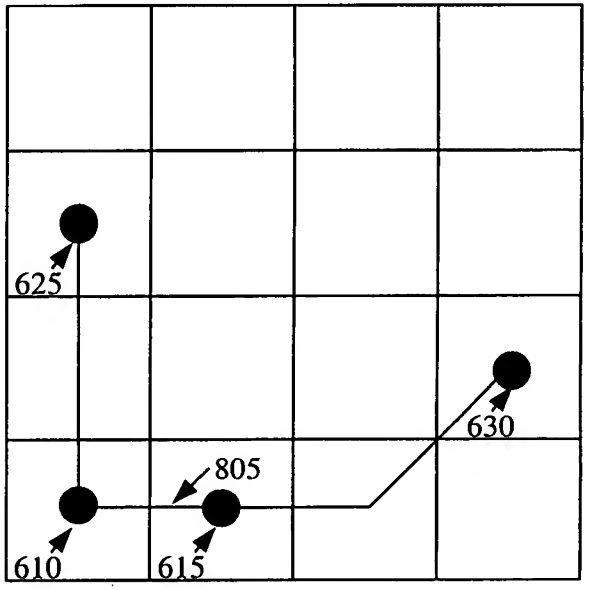

Figure 8

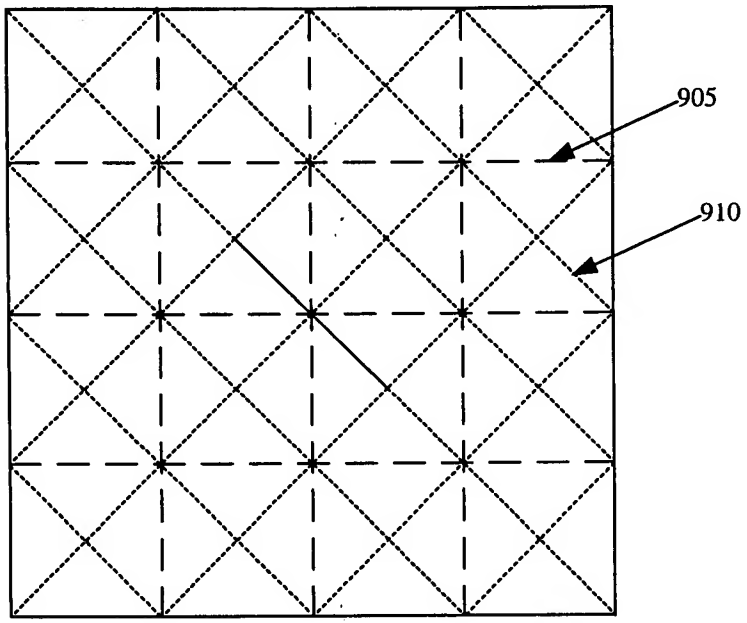

Figure 9

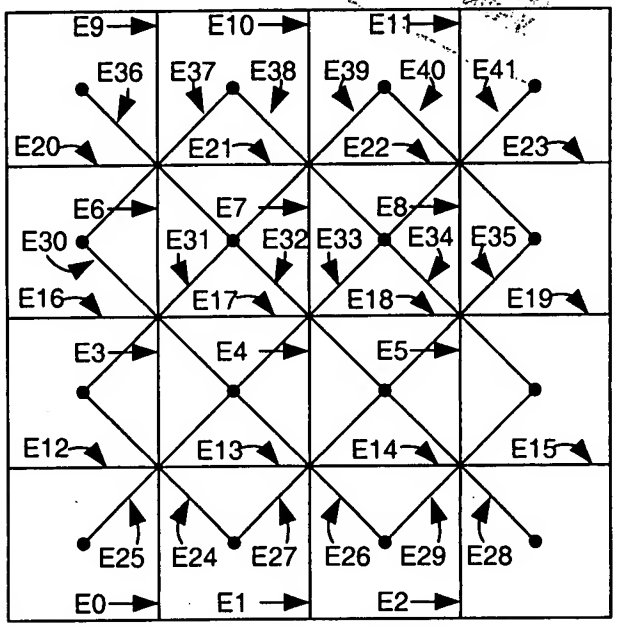

Figure 10

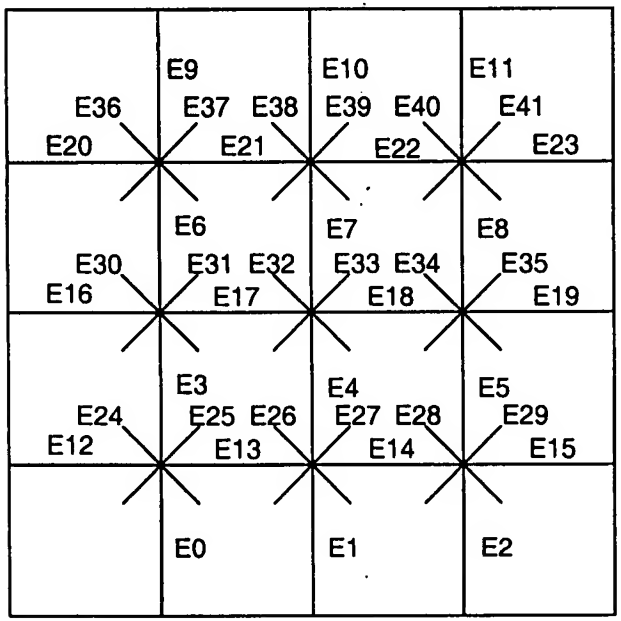

Figure 11

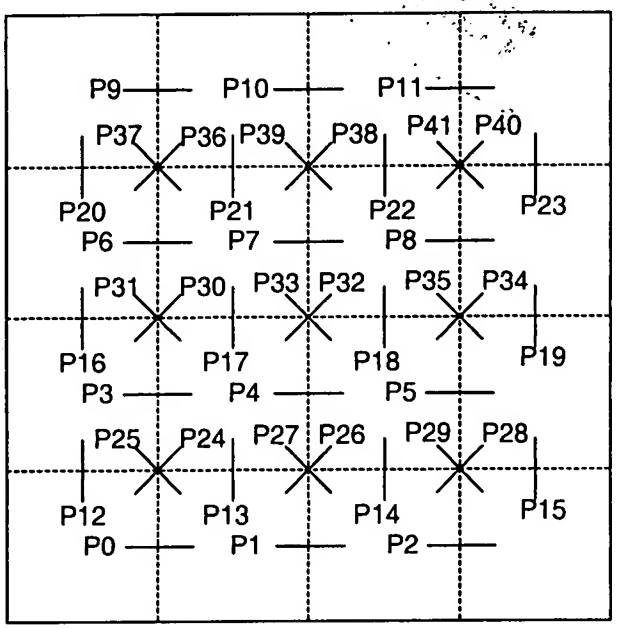

Figure 12

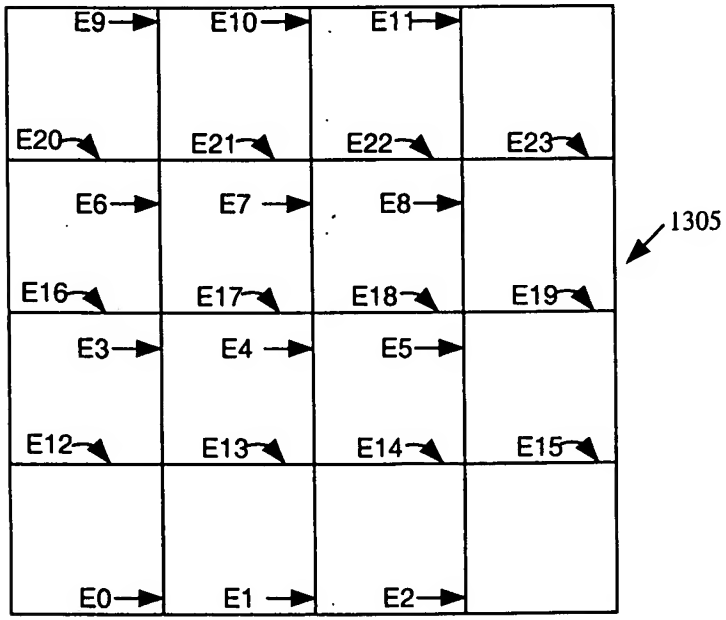

Figure 13

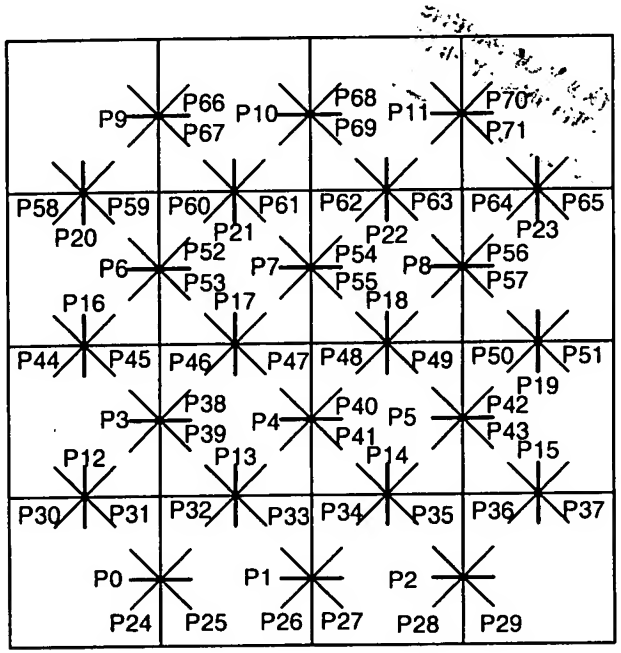

Figure 14

Figure 15

Figure 17

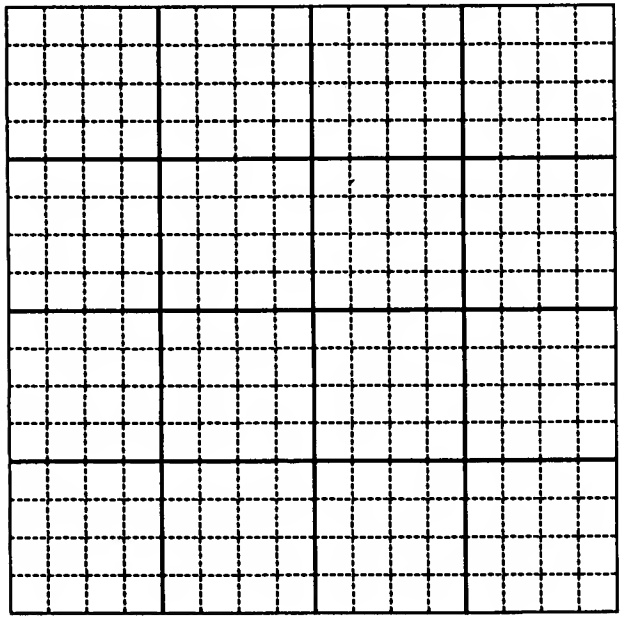

Figure 16

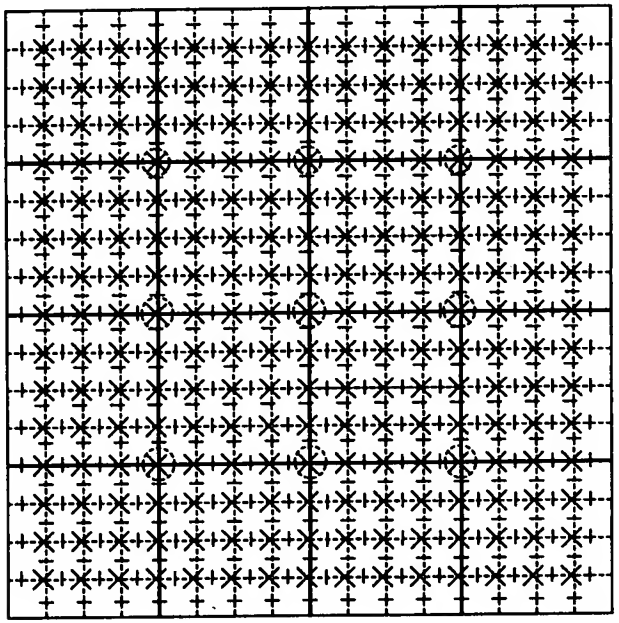

Figure 18

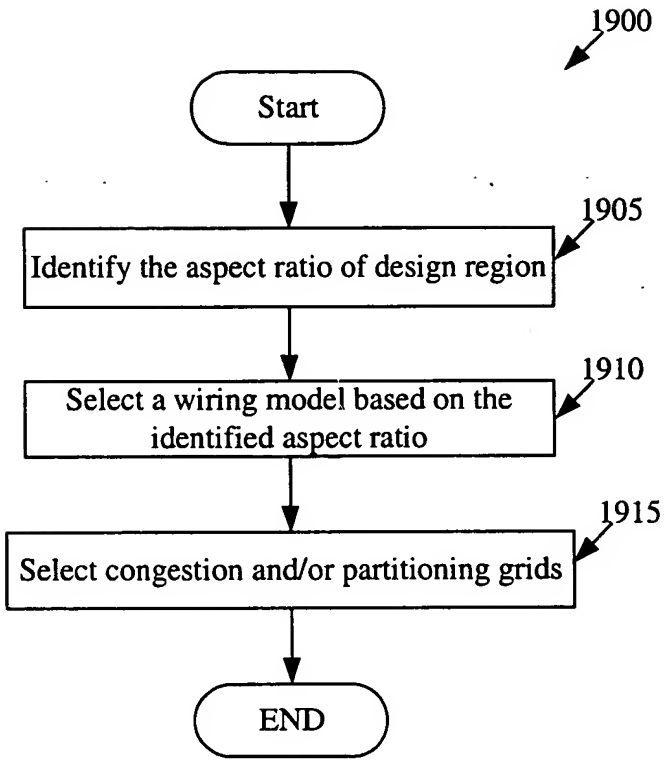
1900
Start
1905
Identify the aspect ratio of design region
1910
Select a wiring model based on the identified aspect ratio
1915
Select congestion and/or partitioning grids
END

Figure 19

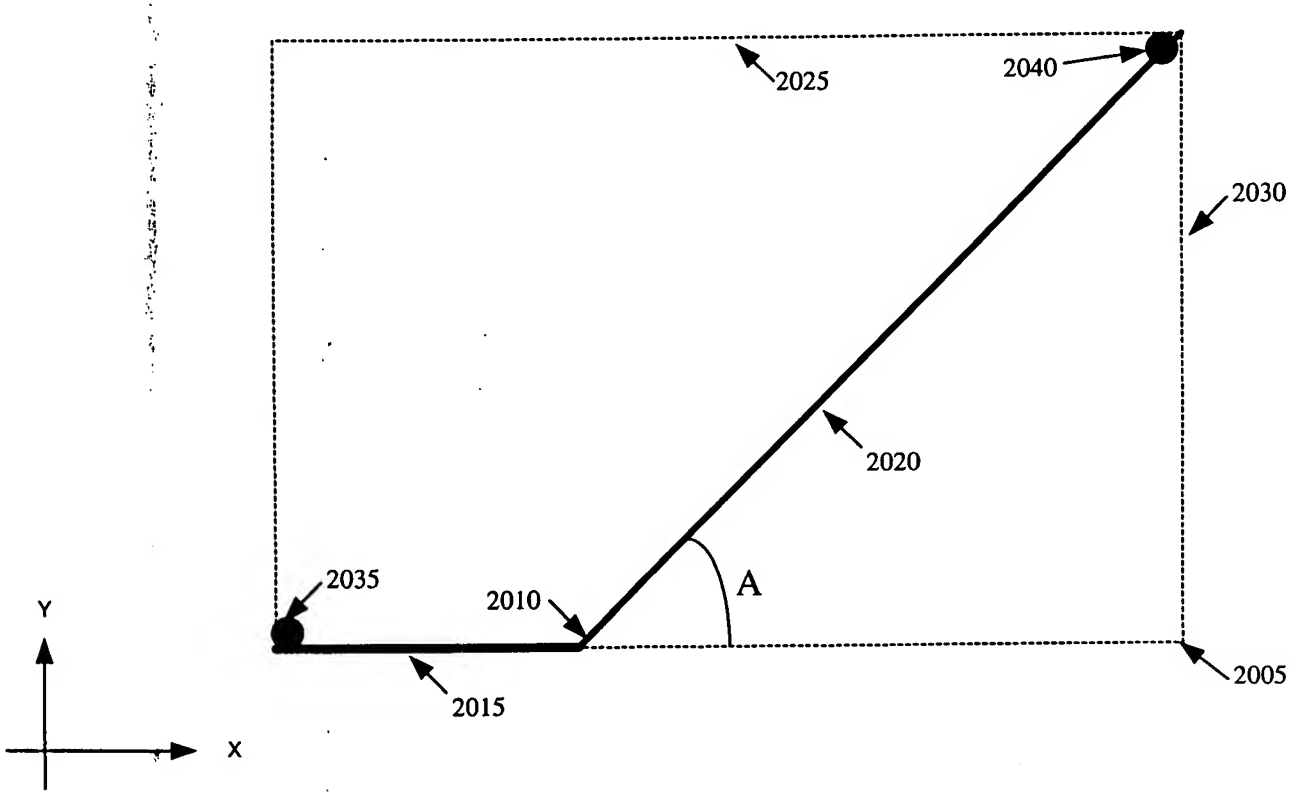

Figure 20

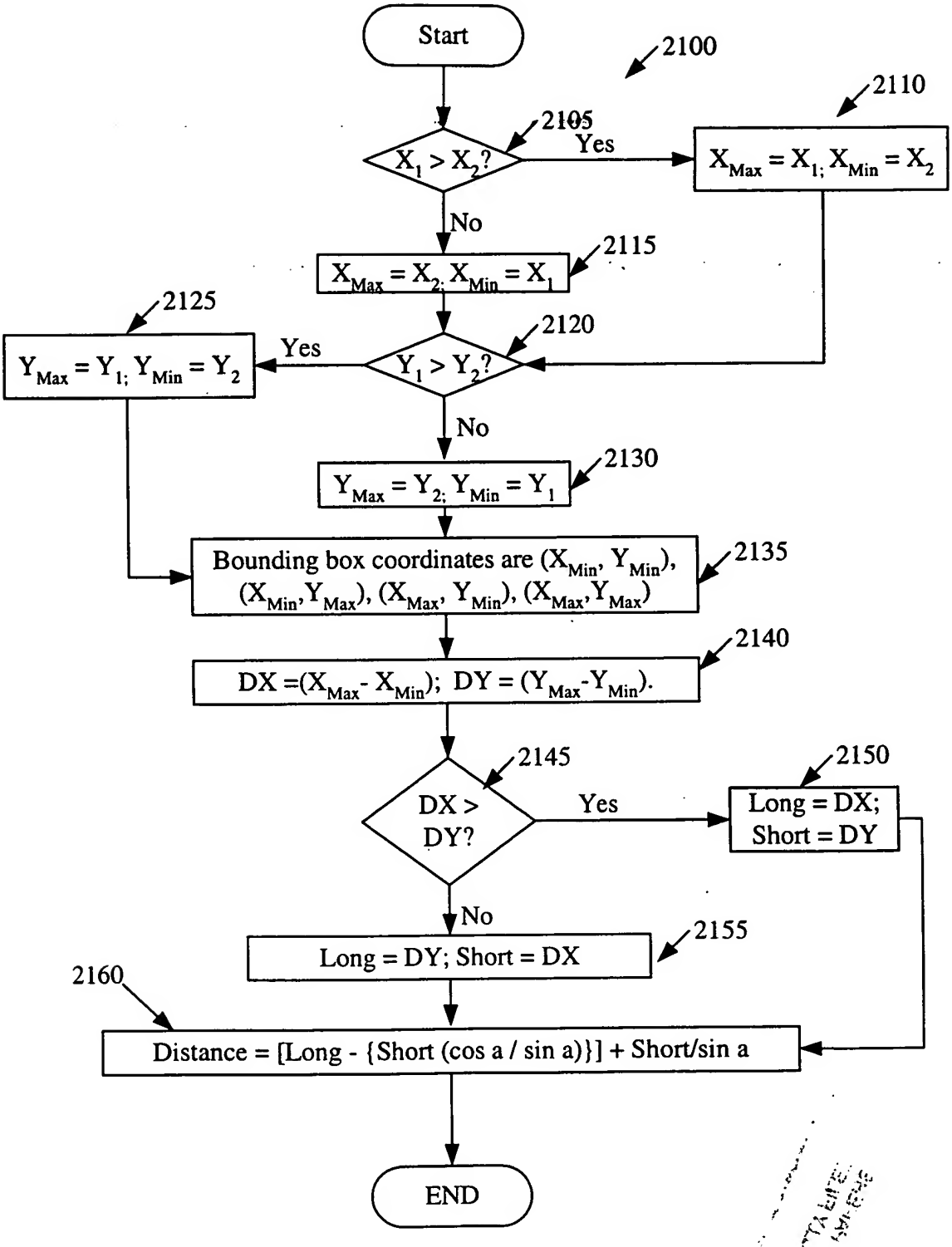
Start
2100
2110
2105
$X_1 > X_2$?
Yes
$X_{Max} = X_1$; $X_{Min} = X_2$
No
2115
$X_{Max} = X_2$; $X_{Min} = X_1$
2125
2120
$Y_{Max} = Y_1$; $Y_{Min} = Y_2$
Yes
$Y_1 > Y_2$?
No
2130
$Y_{Max} = Y_2$; $Y_{Min} = Y_1$
Bounding box coordinates are (X_{Min}, Y_{Min}), (X_{Min},Y_{Max}), (X_{Max}, Y_{Min}), (X_{Max},Y_{Max})
2135
2140
DX =(X_{Max}- X_{Min}); DY = (Y_{Max}-Y_{Min}).
2145
2150
DX > DY?
Yes
Long = DX; Short = DY
No
2155
Long = DY; Short = DX
2160
Distance = [Long - {Short (cos a / sin a)}] + Short/sin a
END

Figure 21

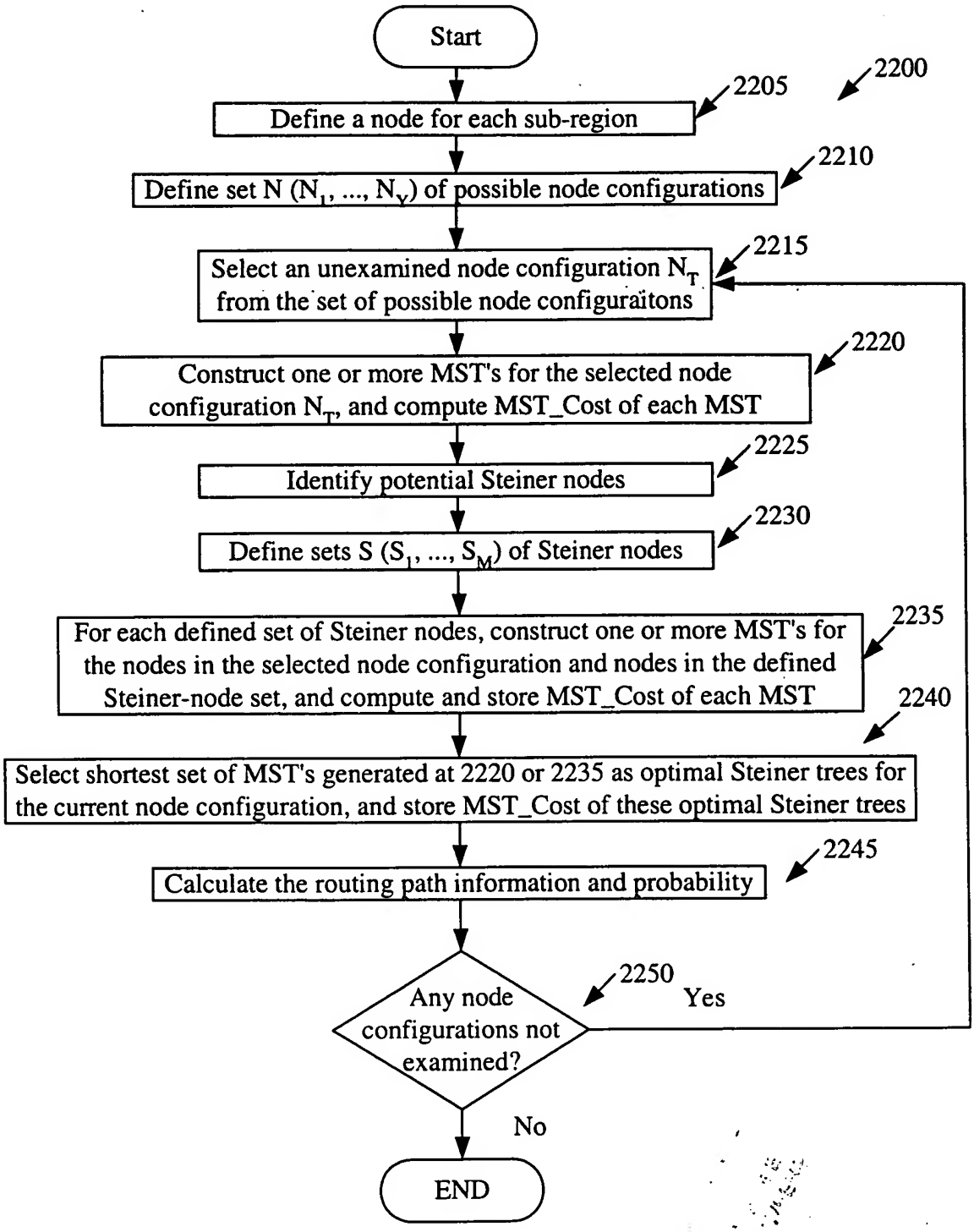
Start
2200
2205
Define a node for each sub-region
2210
Define set N (N_1, ..., N_Y) of possible node configurations
2215
Select an unexamined node configuration N_T from the set of possible node configuraitons
2220
Construct one or more MST's for the selected node configuration N_T, and compute MST_Cost of each MST
2225
Identify potential Steiner nodes
2230
Define sets S (S_1, ..., S_M) of Steiner nodes
2235
For each defined set of Steiner nodes, construct one or more MST's for the nodes in the selected node configuration and nodes in the defined Steiner-node set, and compute and store MST_Cost of each MST
2240
Select shortest set of MST's generated at 2220 or 2235 as optimal Steiner trees for the current node configuration, and store MST_Cost of these optimal Steiner trees
2245
Calculate the routing path information and probability
2250
Any node configurations not examined?
Yes
No
END

Figure 22

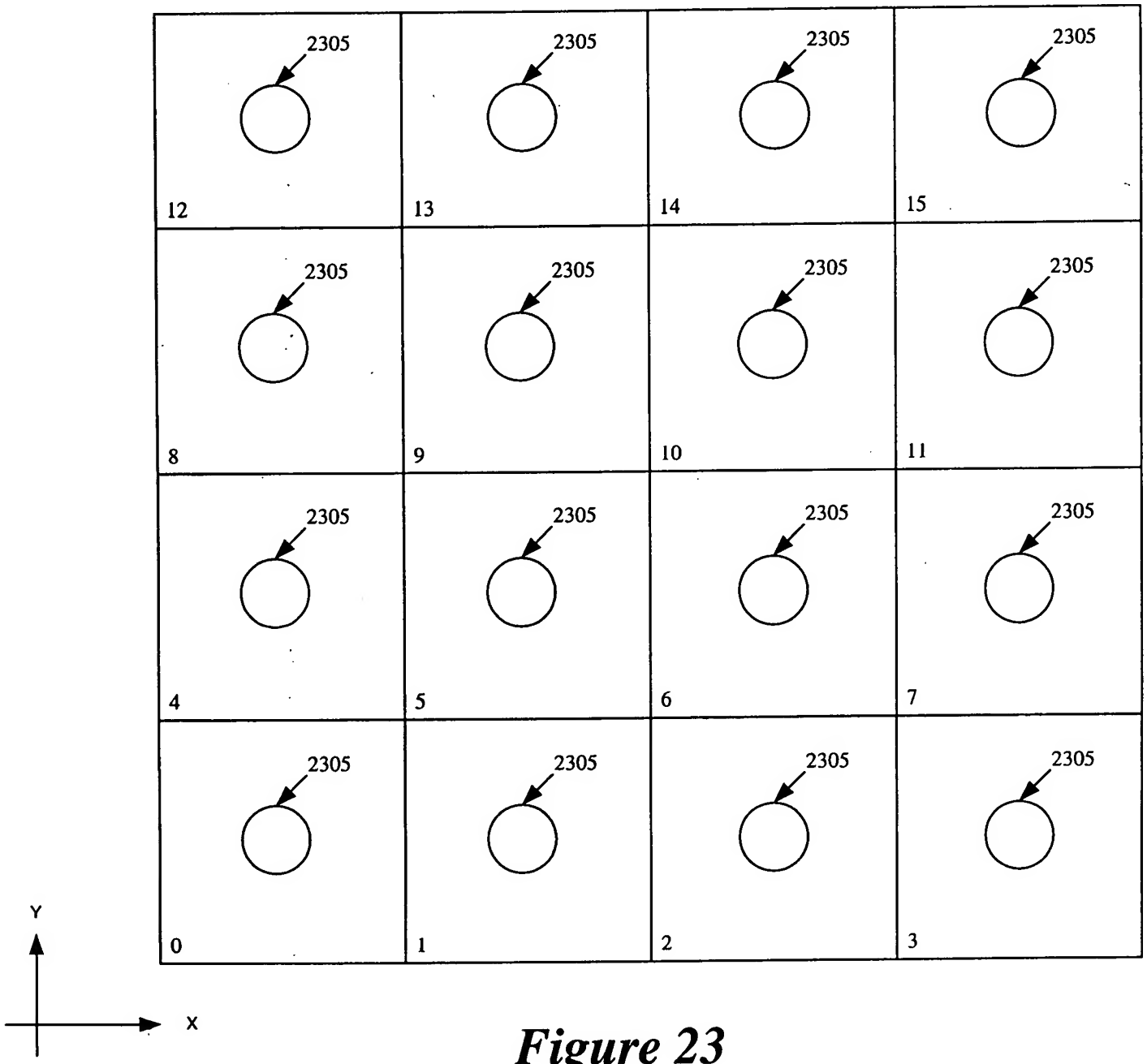

Figure 23

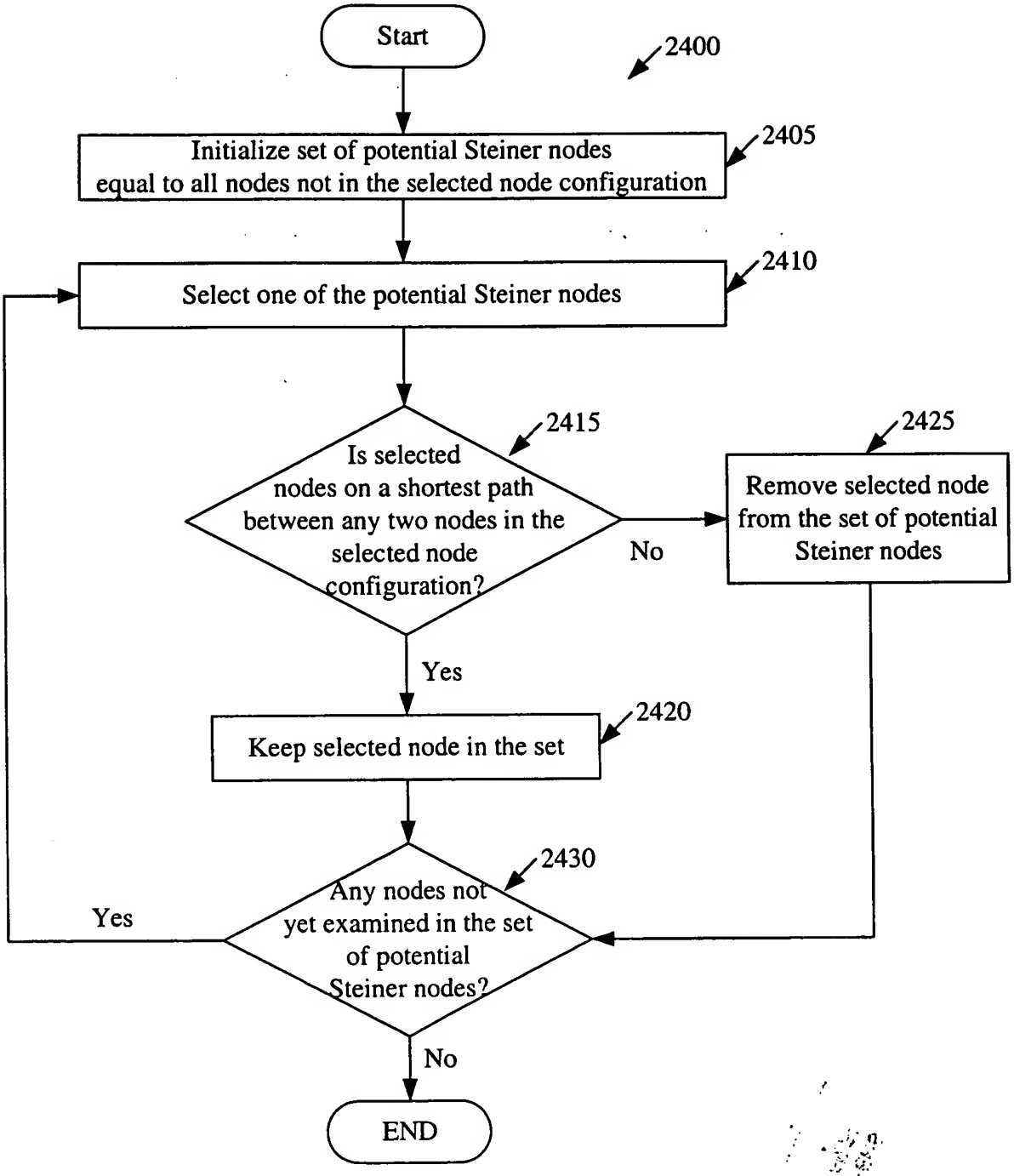
Start
2400
Initialize set of potential Steiner nodes equal to all nodes not in the selected node configuration
2405
Select one of the potential Steiner nodes
2410
Is selected nodes on a shortest path between any two nodes in the selected node configuration?
2415
No
Remove selected node from the set of potential Steiner nodes
2425
Yes
Keep selected node in the set
2420
Any nodes not yet examined in the set of potential Steiner nodes?
2430
Yes
No
END

Figure 24

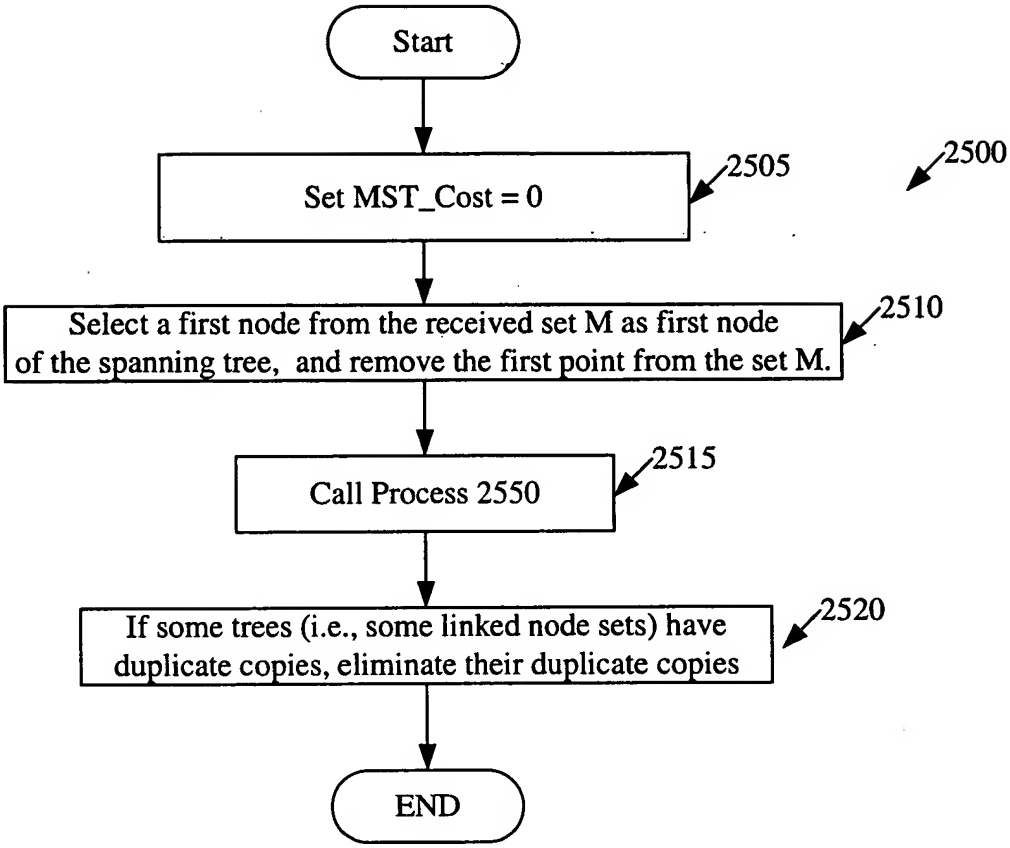
Start
Set MST_Cost = 0
2505
2500
Select a first node from the received set M as first node of the spanning tree, and remove the first point from the set M.
2510
Call Process 2550
2515
If some trees (i.e., some linked node sets) have duplicate copies, eliminate their duplicate copies
2520
END

Figure 25A

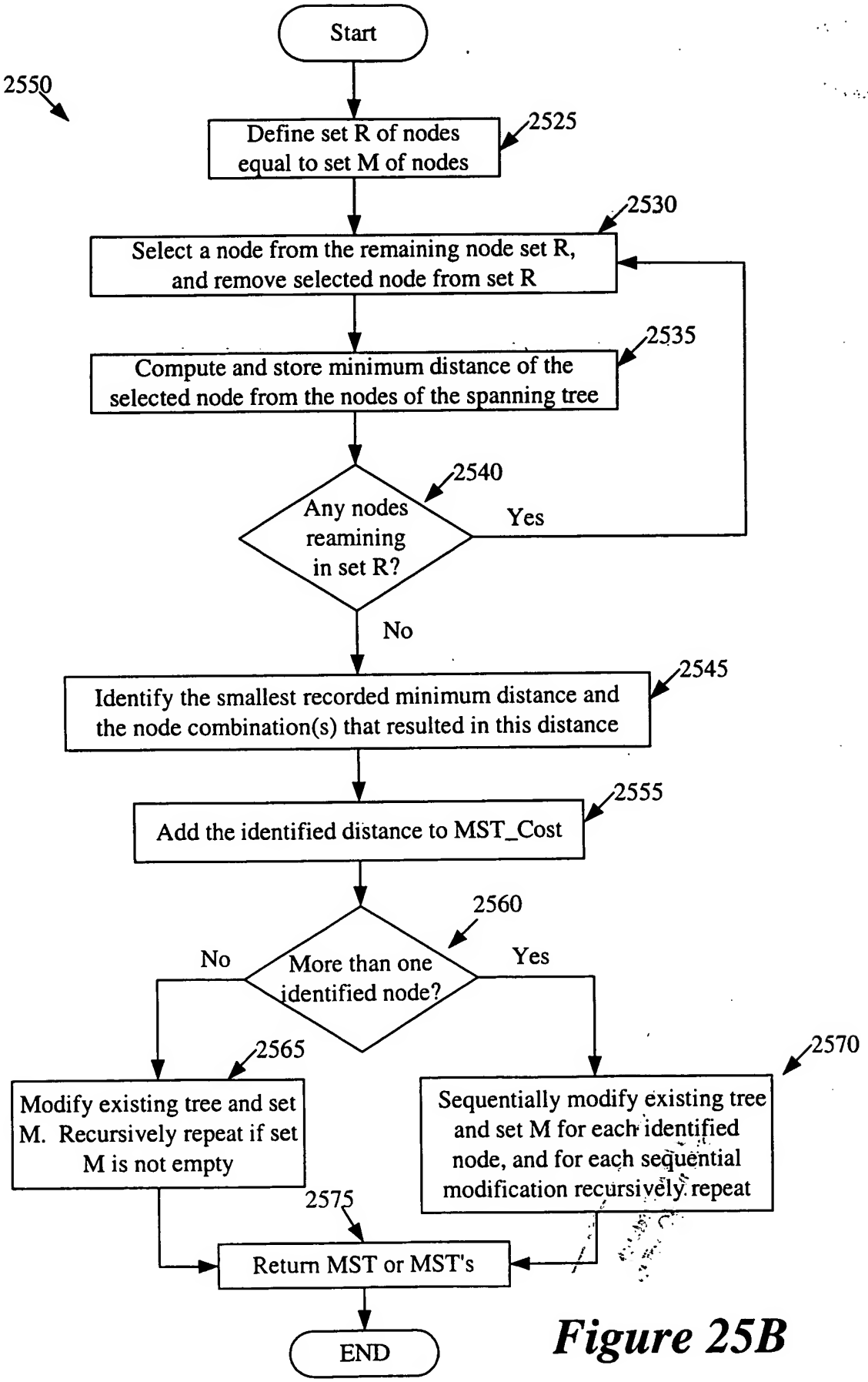
Start
2550
Define set R of nodes equal to set M of nodes
2525
2530
Select a node from the remaining node set R, and remove selected node from set R
2535
Compute and store minimum distance of the selected node from the nodes of the spanning tree
2540
Any nodes reamining in set R?
Yes
No
2545
Identify the smallest recorded minimum distance and the node combination(s) that resulted in this distance
2555
Add the identified distance to MST_Cost
2560
More than one identified node?
No
Yes
2565
Modify existing tree and set M. Recursively repeat if set M is not empty
2570
Sequentially modify existing tree and set M for each identified node, and for each sequential modification recursively repeat
2575
Return MST or MST's
END

Figure 25B

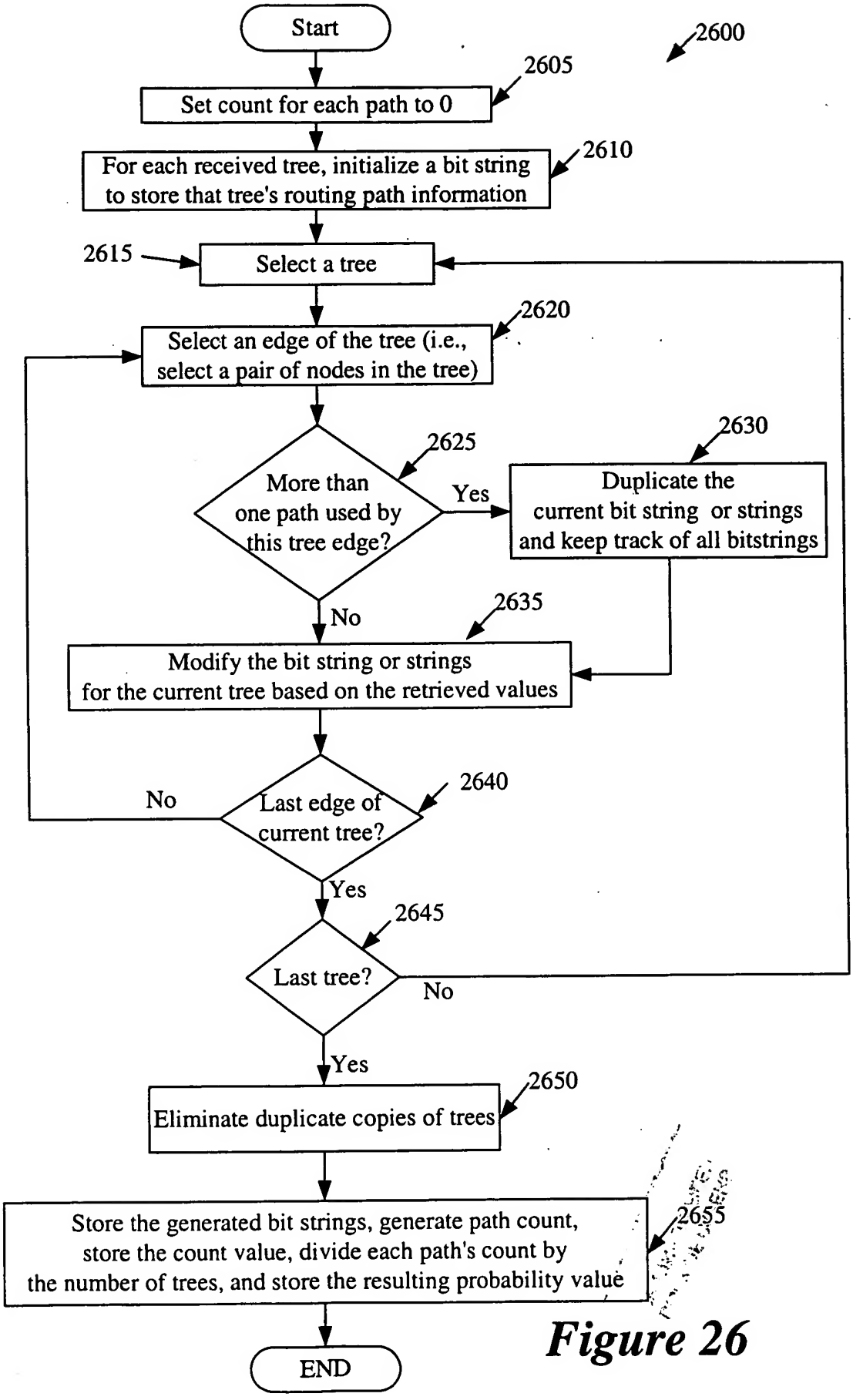
Start
2600
2605
Set count for each path to 0
2610
For each received tree, initialize a bit string to store that tree's routing path information
2615
Select a tree
2620
Select an edge of the tree (i.e., select a pair of nodes in the tree)
2625
More than one path used by this tree edge?
Yes
2630
Duplicate the current bit string or strings and keep track of all bitstrings
No
2635
Modify the bit string or strings for the current tree based on the retrieved values
2640
Last edge of current tree?
No
Yes
2645
Last tree?
No
Yes
2650
Eliminate duplicate copies of trees
2655
Store the generated bit strings, generate path count, store the count value, divide each path's count by the number of trees, and store the resulting probability value
END

Figure 26

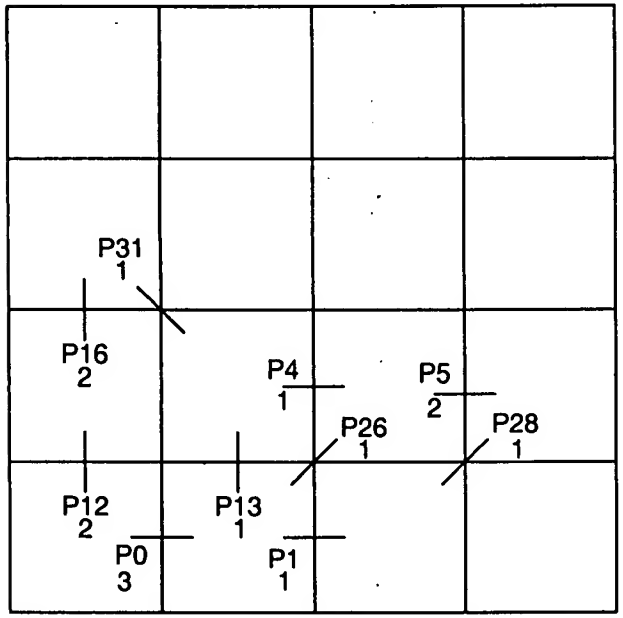

Figure 27

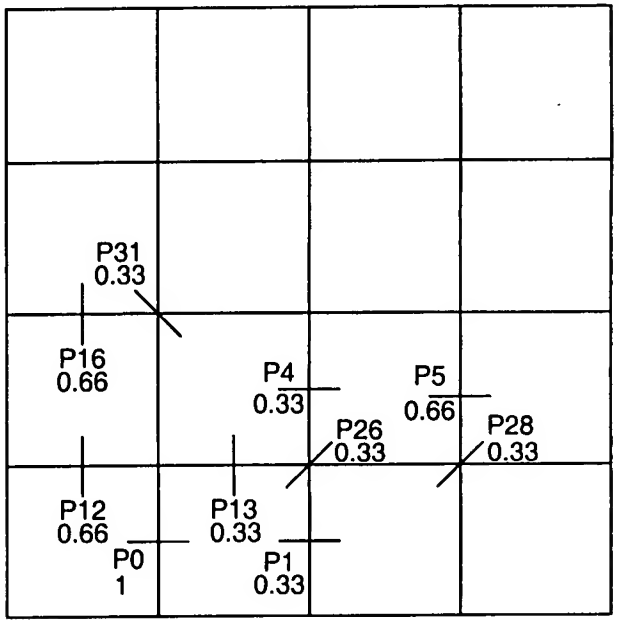

Figure 28

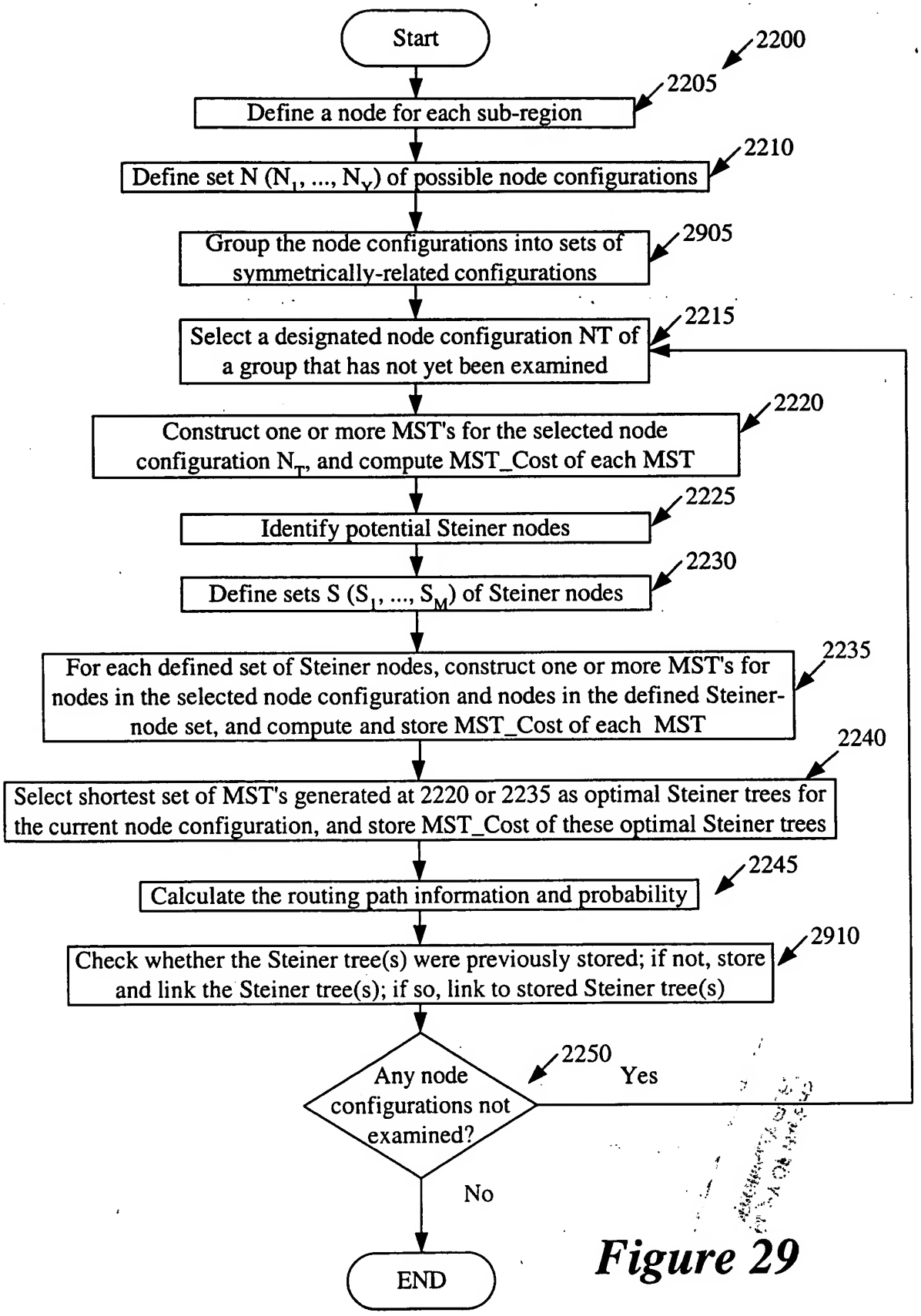
Start
2200
2205
Define a node for each sub-region
2210
Define set N (N_1, ..., N_Y) of possible node configurations
2905
Group the node configurations into sets of symmetrically-related configurations
2215
Select a designated node configuration NT of a group that has not yet been examined
2220
Construct one or more MST's for the selected node configuration N_T, and compute MST_Cost of each MST
2225
Identify potential Steiner nodes
2230
Define sets S (S_1, ..., S_M) of Steiner nodes
2235
For each defined set of Steiner nodes, construct one or more MST's for nodes in the selected node configuration and nodes in the defined Steiner-node set, and compute and store MST_Cost of each MST
2240
Select shortest set of MST's generated at 2220 or 2235 as optimal Steiner trees for the current node configuration, and store MST_Cost of these optimal Steiner trees
2245
Calculate the routing path information and probability
2910
Check whether the Steiner tree(s) were previously stored; if not, store and link the Steiner tree(s); if so, link to stored Steiner tree(s)
2250
Any node configurations not examined?
Yes
No
END

Figure 29

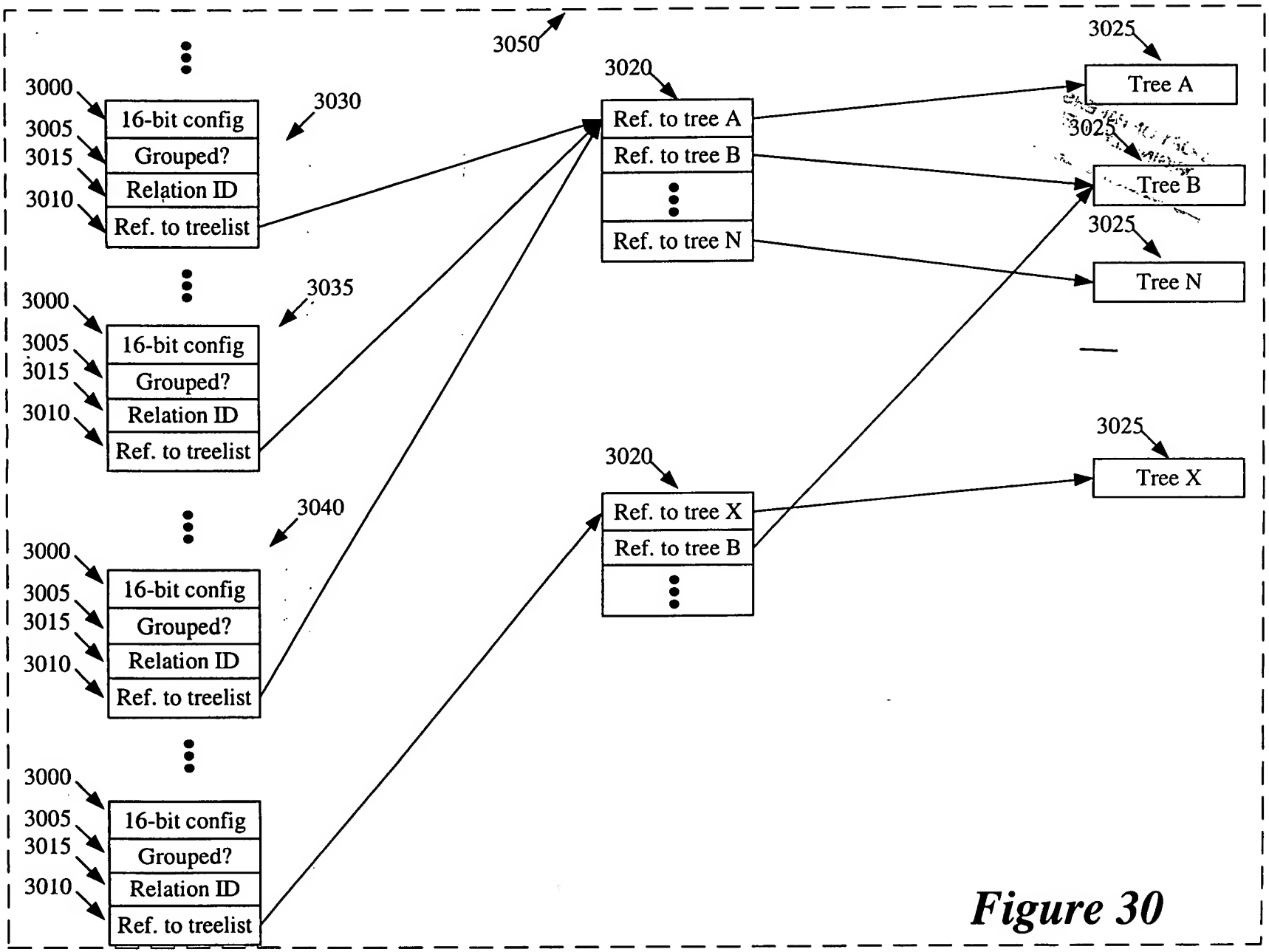

Figure 30

10047076.042502

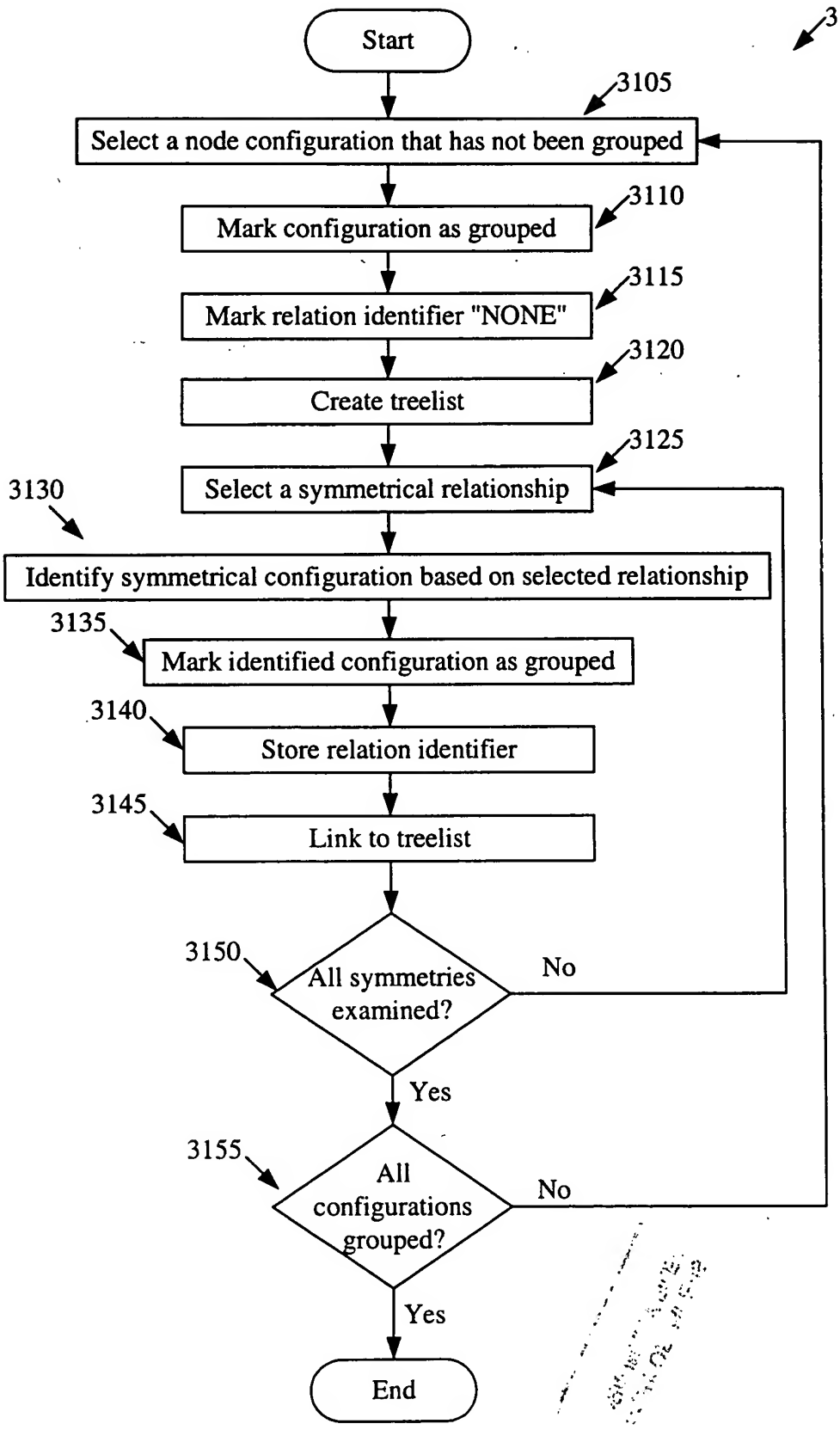
3100
Start
3105
Select a node configuration that has not been grouped
3110
Mark configuration as grouped
3115
Mark relation identifier "NONE"
3120
Create treelist
3125
Select a symmetrical relationship
3130
Identify symmetrical configuration based on selected relationship
3135
Mark identified configuration as grouped
3140
Store relation identifier
3145
Link to treelist
3150
All symmetries examined?
No
Yes
3155
All configurations grouped?
No
Yes
End

Figure 31

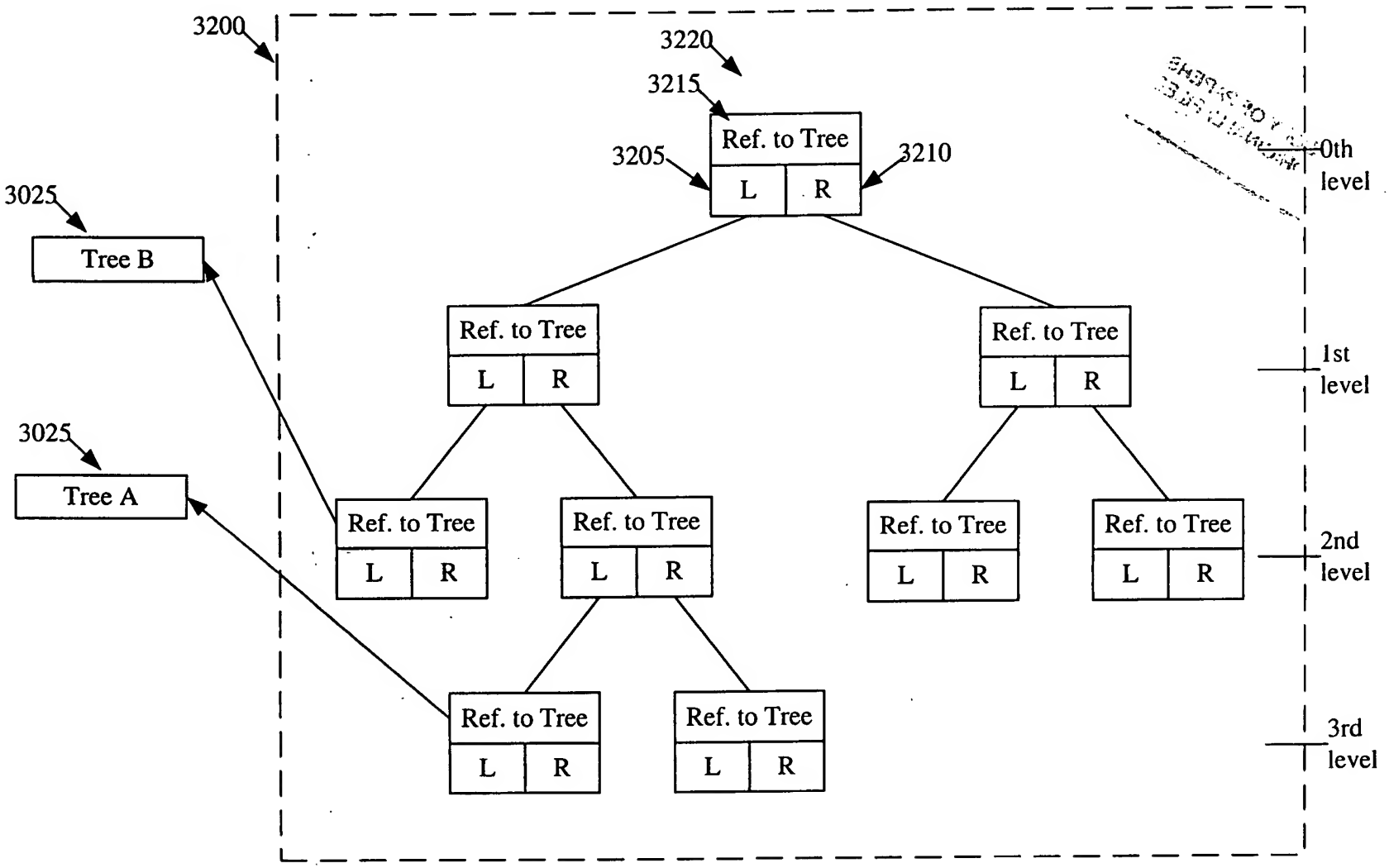
3200
3220
3215
Ref. to Tree
L R
3205
3210
0th level
3025
Tree B
3025
Tree A
Ref. to Tree
L R
1st level
2nd level
3rd level

Figure 32

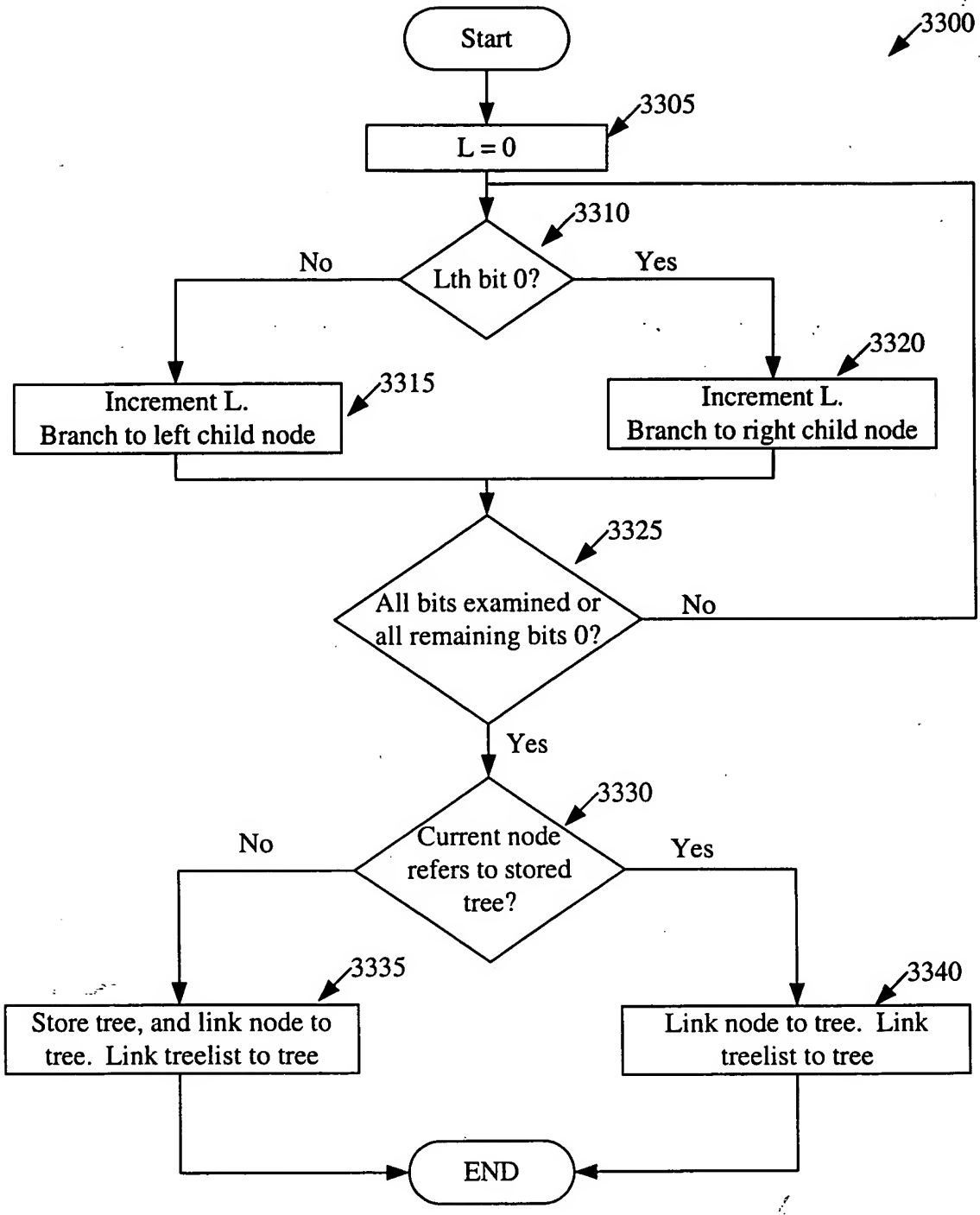
Start
3300
3305
L = 0
3310
Lth bit 0?
No
Yes
3315
Increment L.
Branch to left child node
3320
Increment L.
Branch to right child node
3325
All bits examined or
all remaining bits 0?
No
Yes
3330
Current node
refers to stored
tree?
No
Yes
3335
Store tree, and link node to
tree. Link treelist to tree
3340
Link node to tree. Link
treelist to tree
END

Figure 33

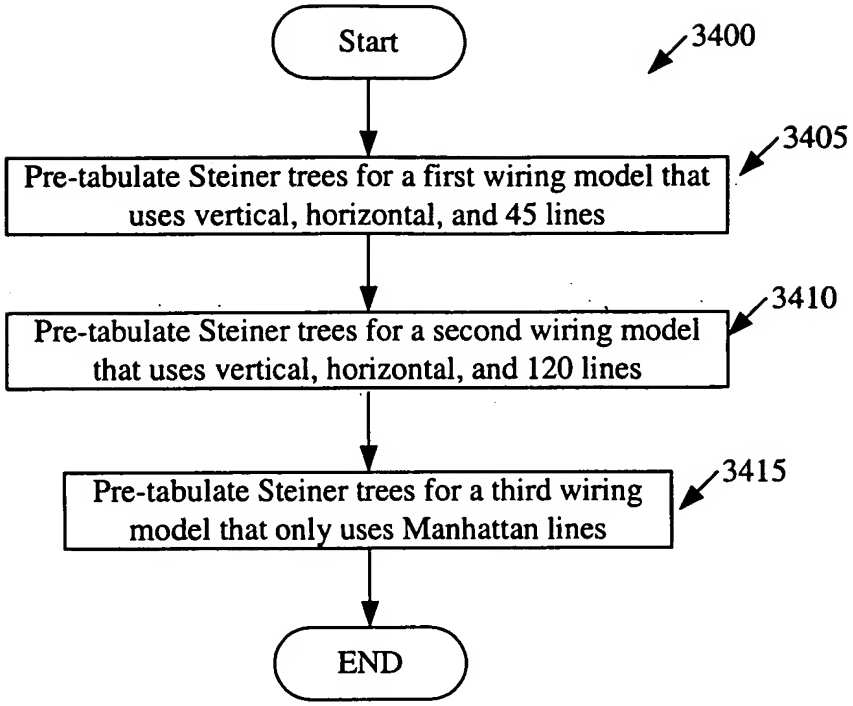
Start
3400
Pre-tabulate Steiner trees for a first wiring model that uses vertical, horizontal, and 45 lines
3405
Pre-tabulate Steiner trees for a second wiring model that uses vertical, horizontal, and 120 lines
3410
Pre-tabulate Steiner trees for a third wiring model that only uses Manhattan lines
3415
END

Figure 34

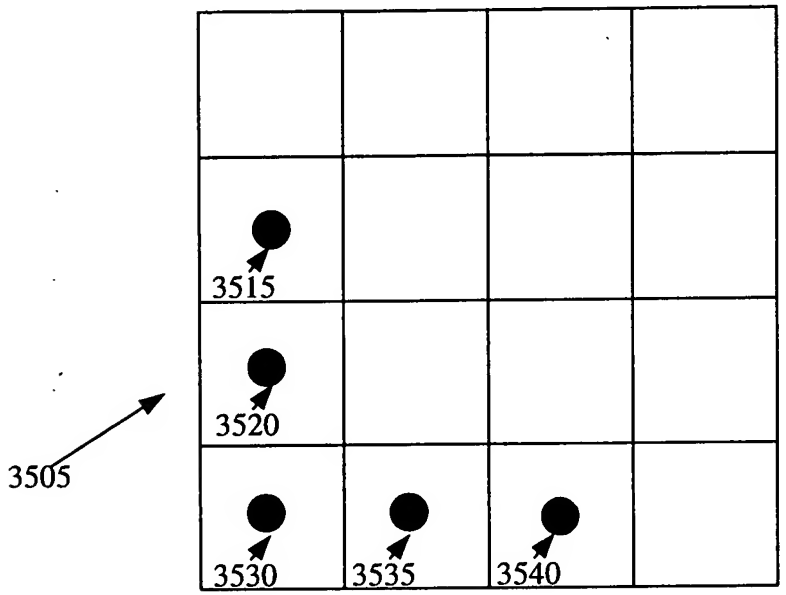

Figure 35A

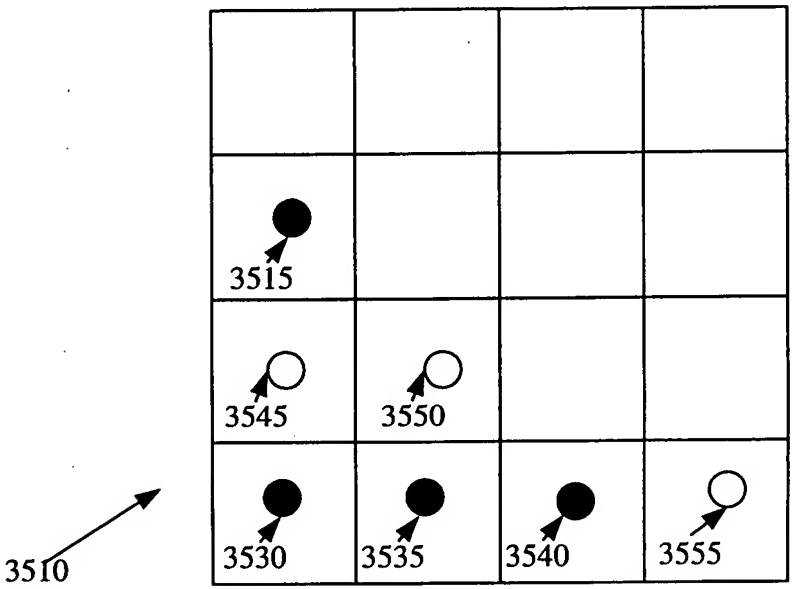

Figure 35B

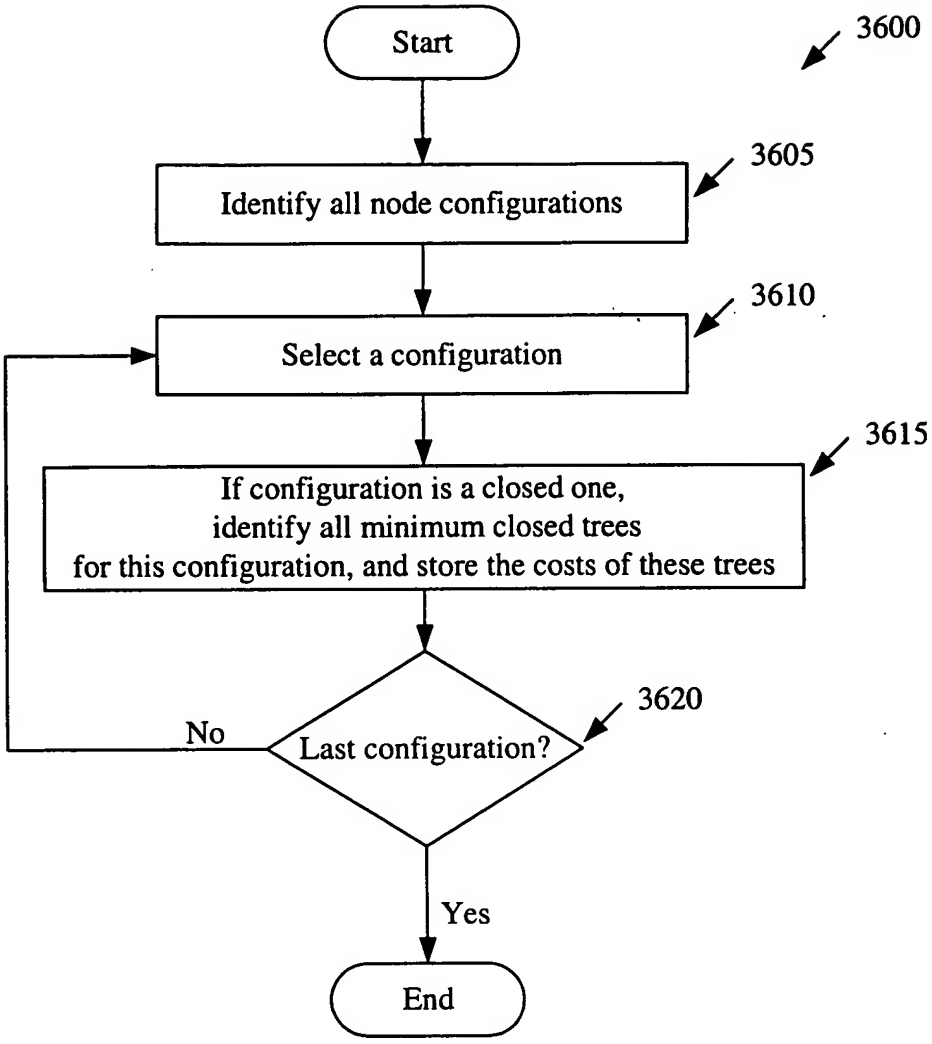
Start
3600
Identify all node configurations
3605
Select a configuration
3610
If configuration is a closed one,
identify all minimum closed trees
for this configuration, and store the costs of these trees
3615
Last configuration?
3620
No
Yes
End

Figure 36

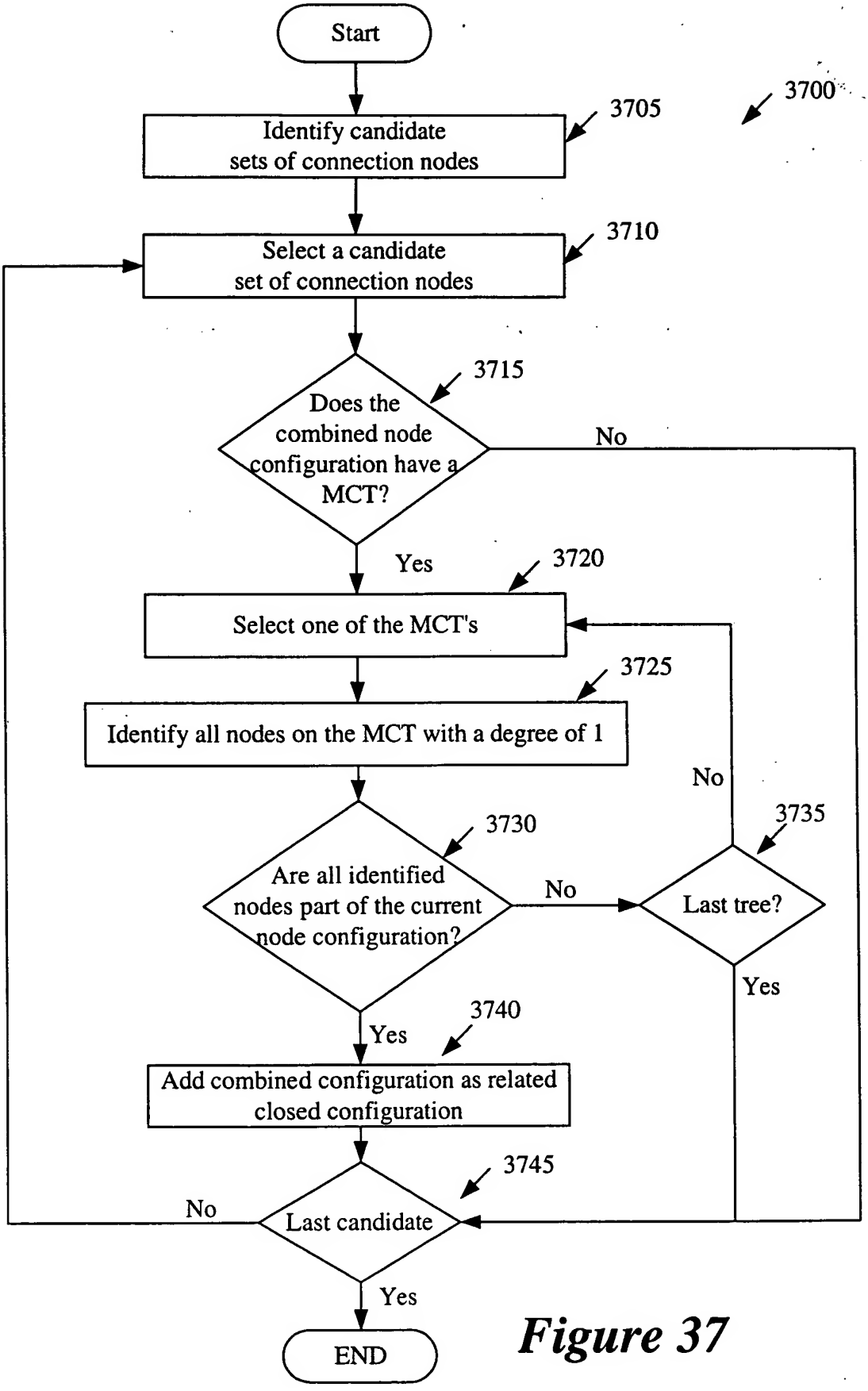
Start
3700
Identify candidate sets of connection nodes
3705
Select a candidate set of connection nodes
3710
Does the combined node configuration have a MCT?
3715
No
Yes
Select one of the MCT's
3720
Identify all nodes on the MCT with a degree of 1
3725
No
Are all identified nodes part of the current node configuration?
3730
No
Last tree?
3735
Yes
Yes
Add combined configuration as related closed configuration
3740
Last candidate
3745
No
Yes
END

Figure 37

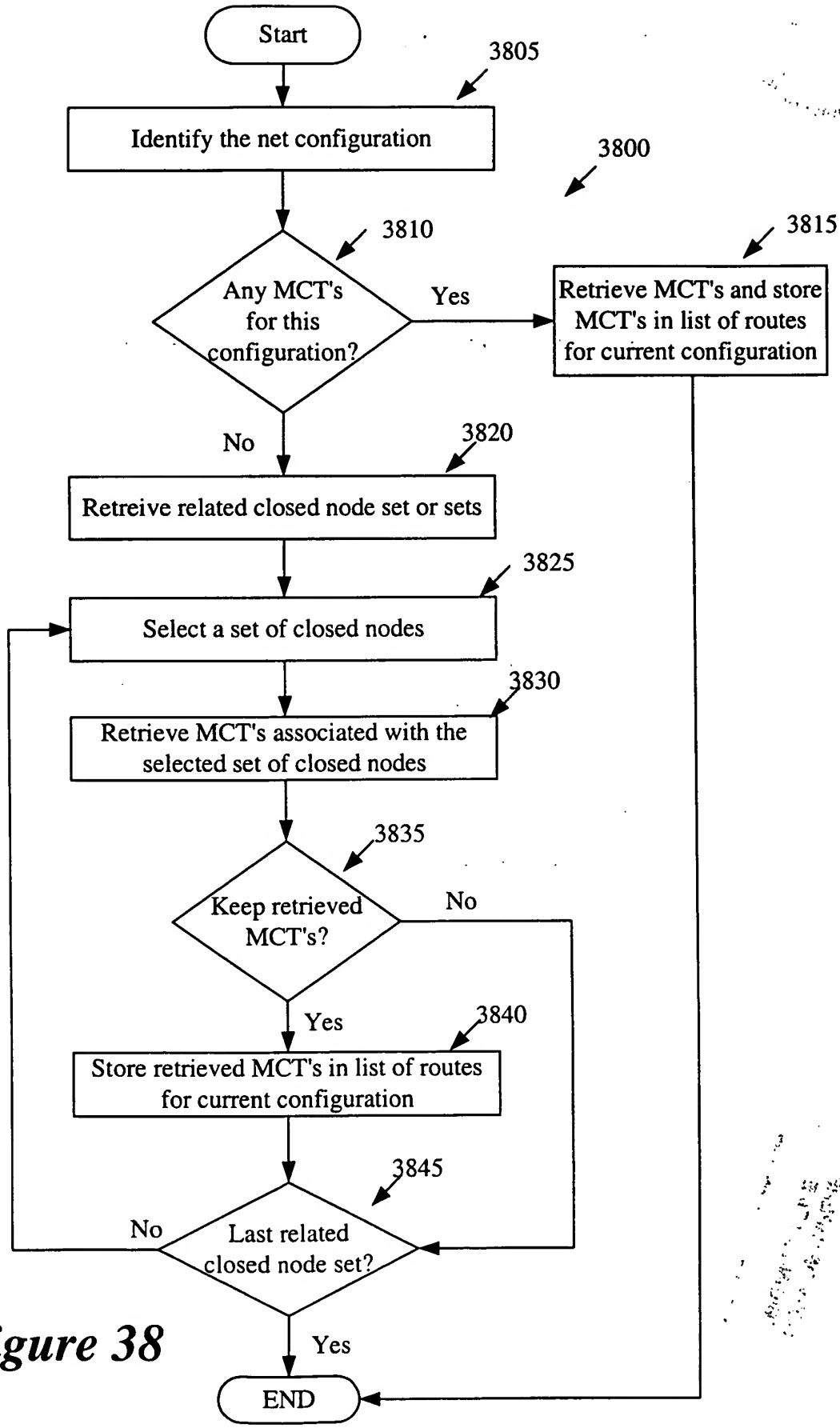
Start
3805
Identify the net configuration
3800
3810
Any MCT's for this configuration?
Yes
3815
Retrieve MCT's and store MCT's in list of routes for current configuration
No
3820
Retreive related closed node set or sets
3825
Select a set of closed nodes
3830
Retrieve MCT's associated with the selected set of closed nodes
3835
Keep retrieved MCT's?
No
Yes
3840
Store retrieved MCT's in list of routes for current configuration
3845
No
Last related closed node set?
Yes
END

Figure 38

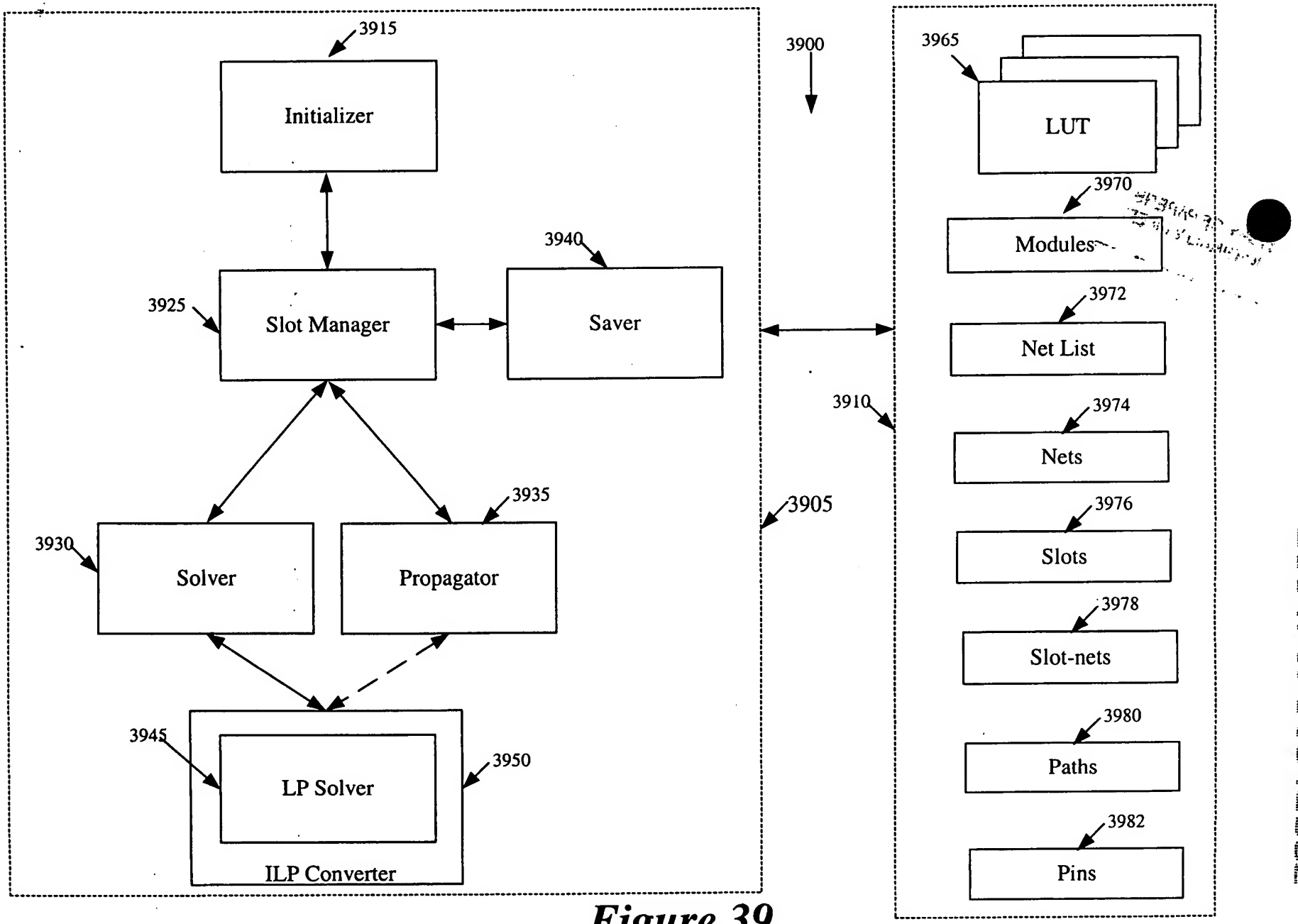
3915
Initializer
3900
3965
LUT
3970
Modules
3940
3925
Slot Manager
Saver
3972
Net List
3910
3974
Nets
3935
3905
3976
Slots
3930
Solver
Propagator
3978
Slot-nets
3980
3945
LP Solver
3950
Paths
3982
ILP Converter
Pins

Figure 39

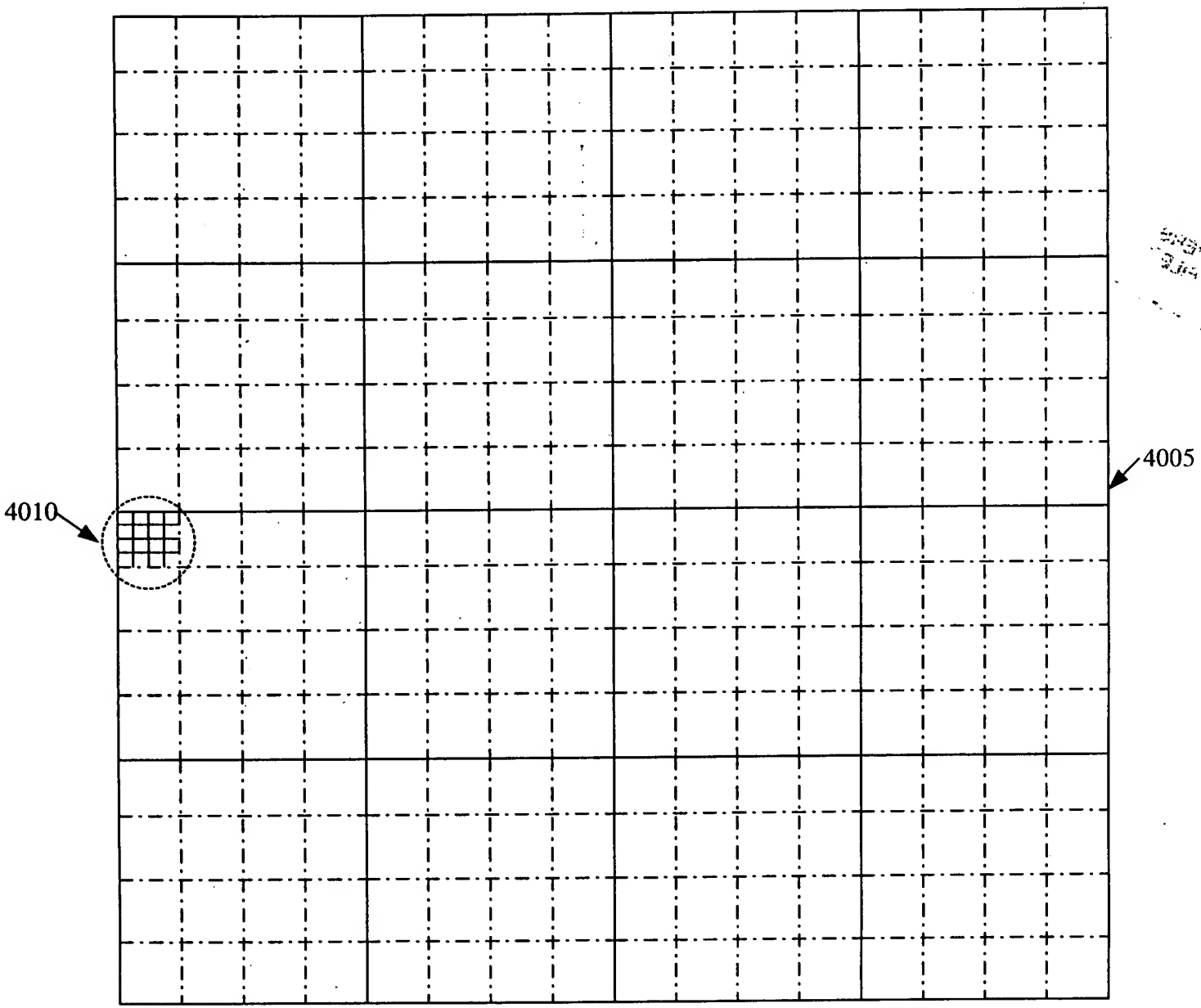

Figure 40

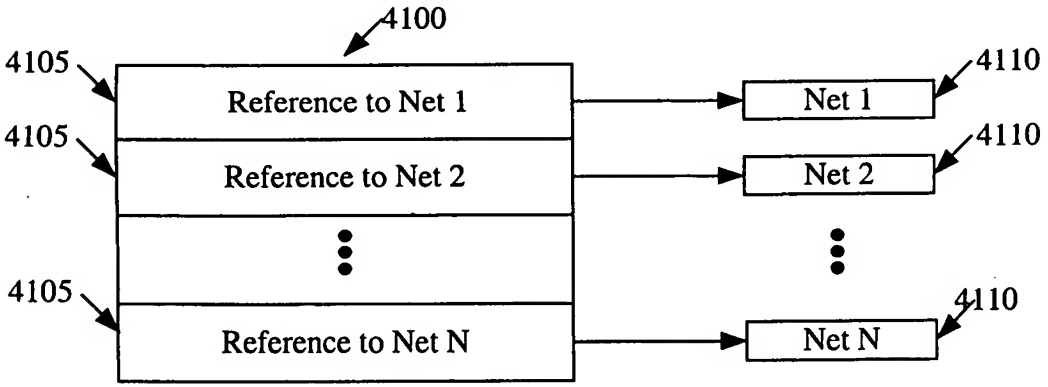

Figure 41

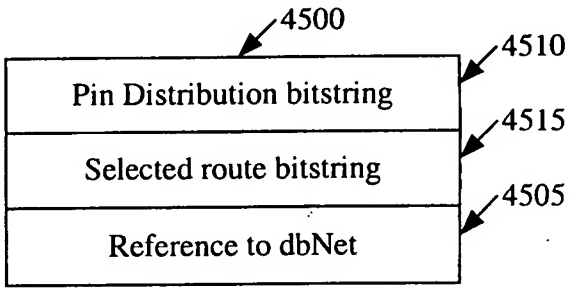

Figure 45

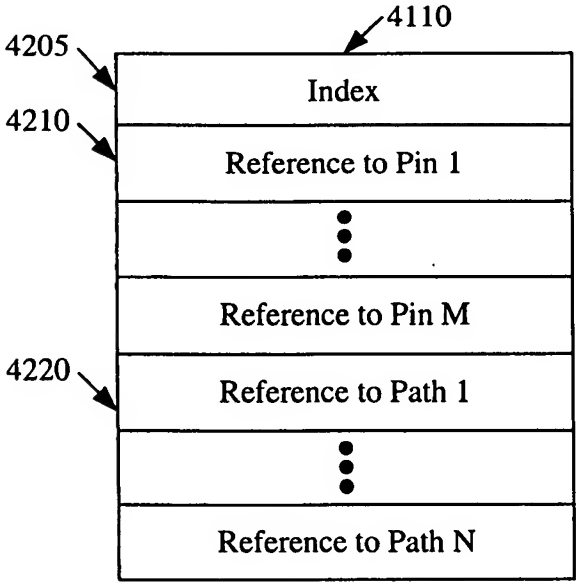

Figure 42

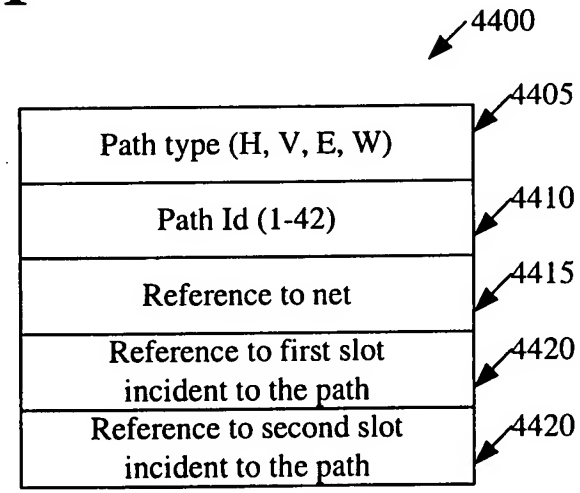

Figure 44

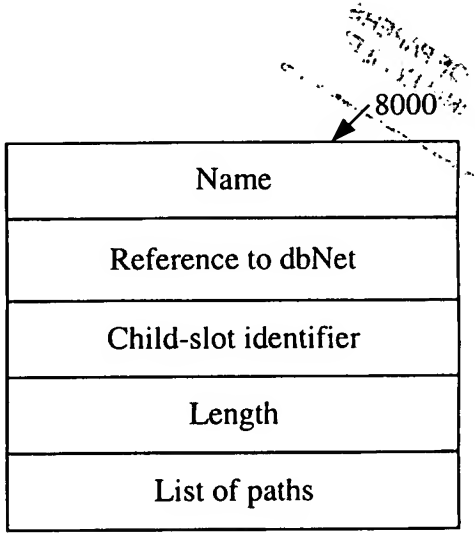

Figure 80

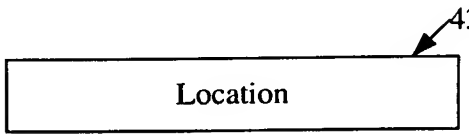

Figure 43

10047976 - 042502

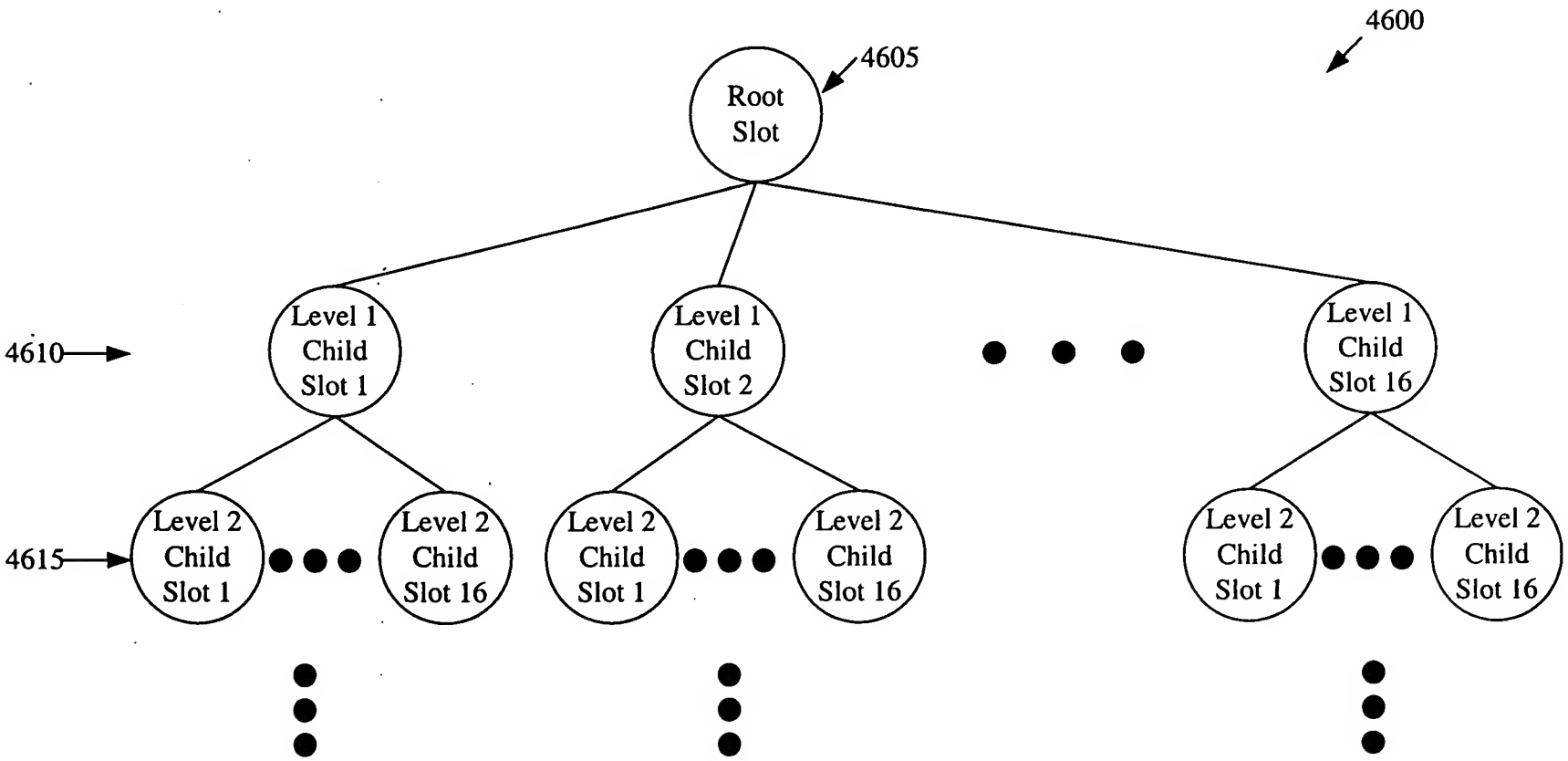
4600
4605
Root
Slot
4610
Level 1
Child
Slot 1
Level 1
Child
Slot 2
Level 1
Child
Slot 16
4615
Level 2
Child
Slot 1
Level 2
Child
Slot 16
Level 2
Child
Slot 1
Level 2
Child
Slot 16
Level 2
Child
Slot 1
Level 2
Child
Slot 16

Figure 46

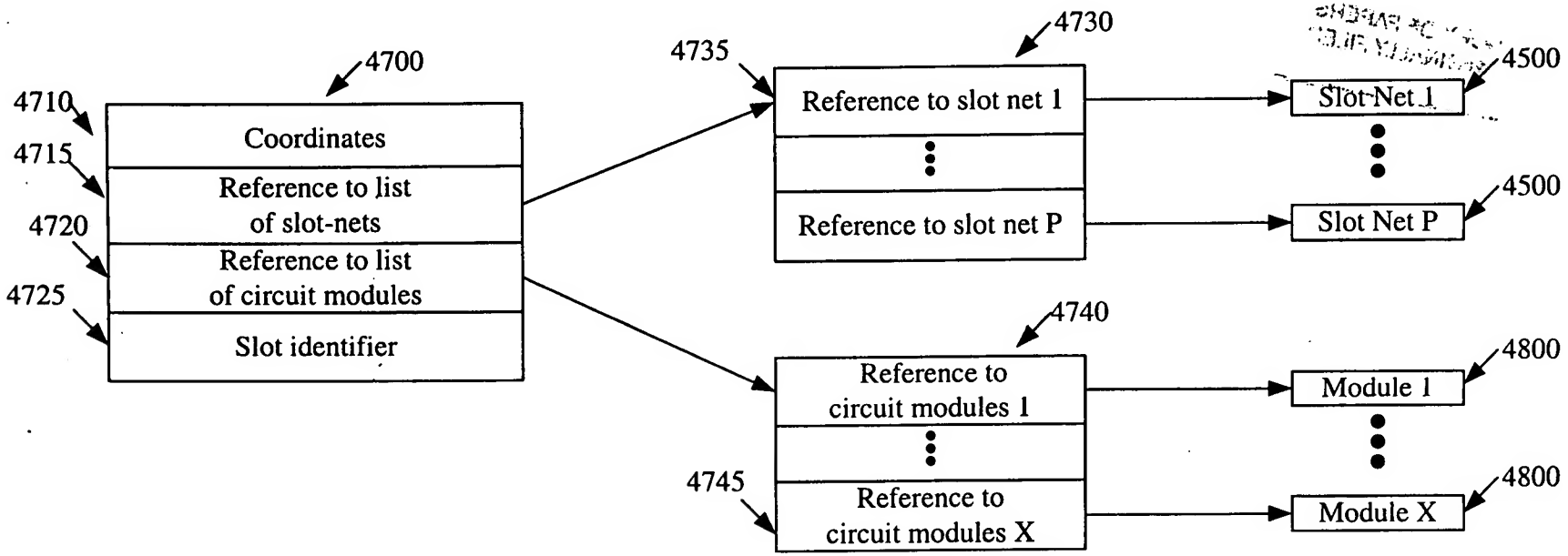

Figure 47

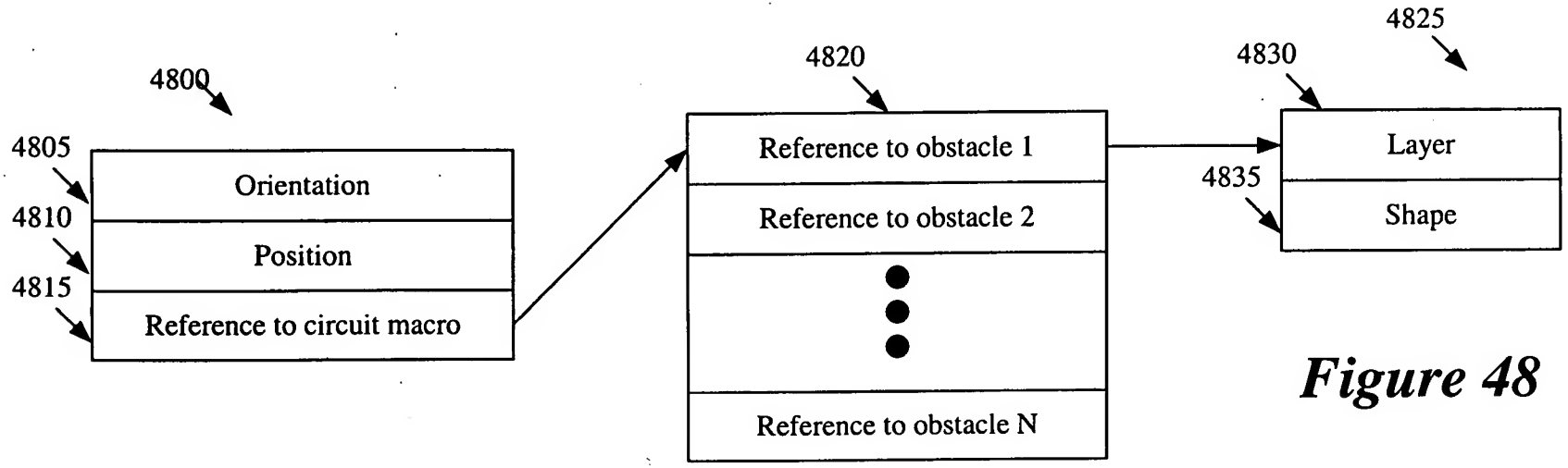

Figure 48

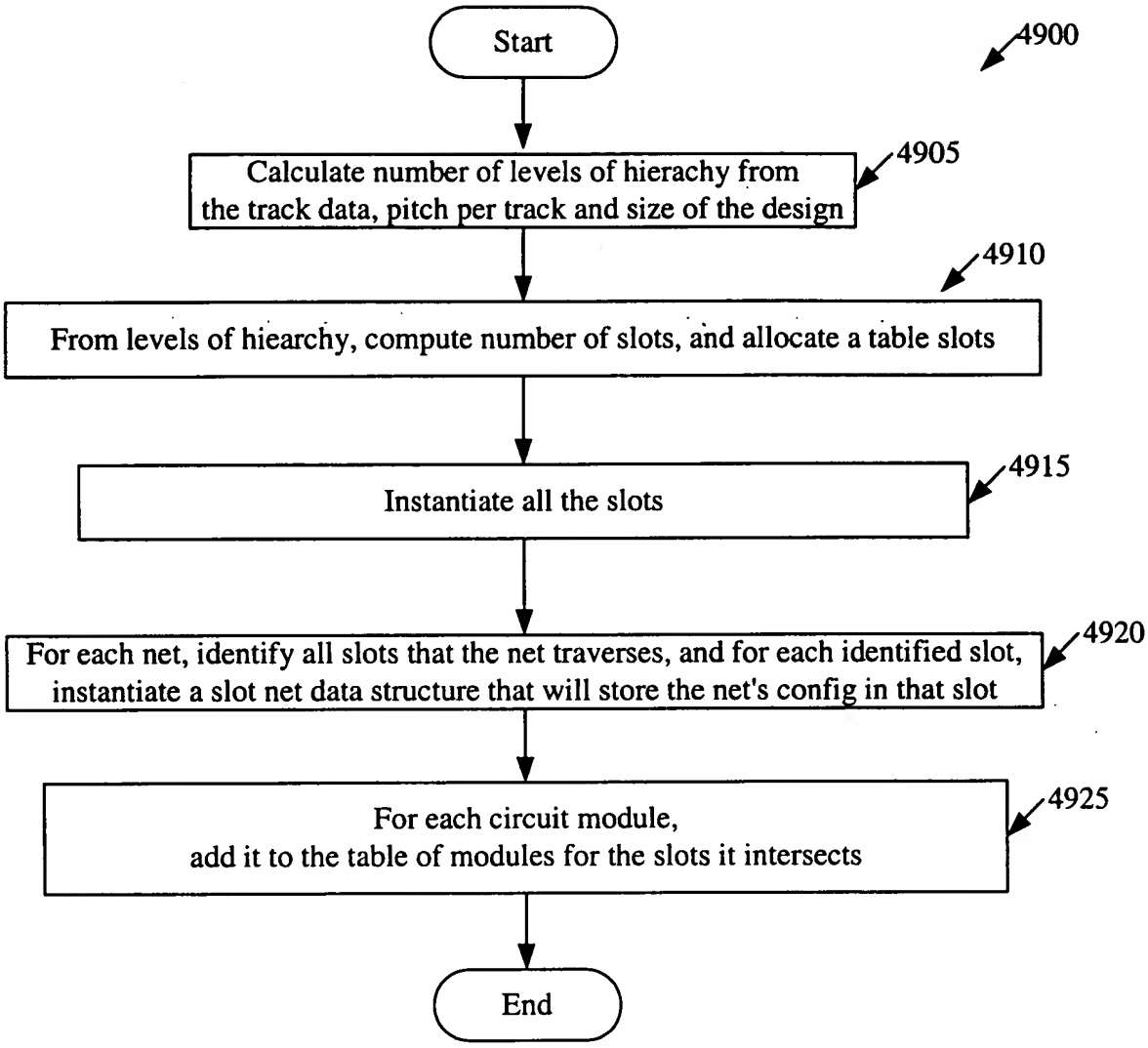
Start
4900
Calculate number of levels of hierachy from the track data, pitch per track and size of the design
4905
From levels of hiearchy, compute number of slots, and allocate a table slots
4910
Instantiate all the slots
4915
For each net, identify all slots that the net traverses, and for each identified slot, instantiate a slot net data structure that will store the net's config in that slot
4920
For each circuit module, add it to the table of modules for the slots it intersects
4925
End

Figure 49

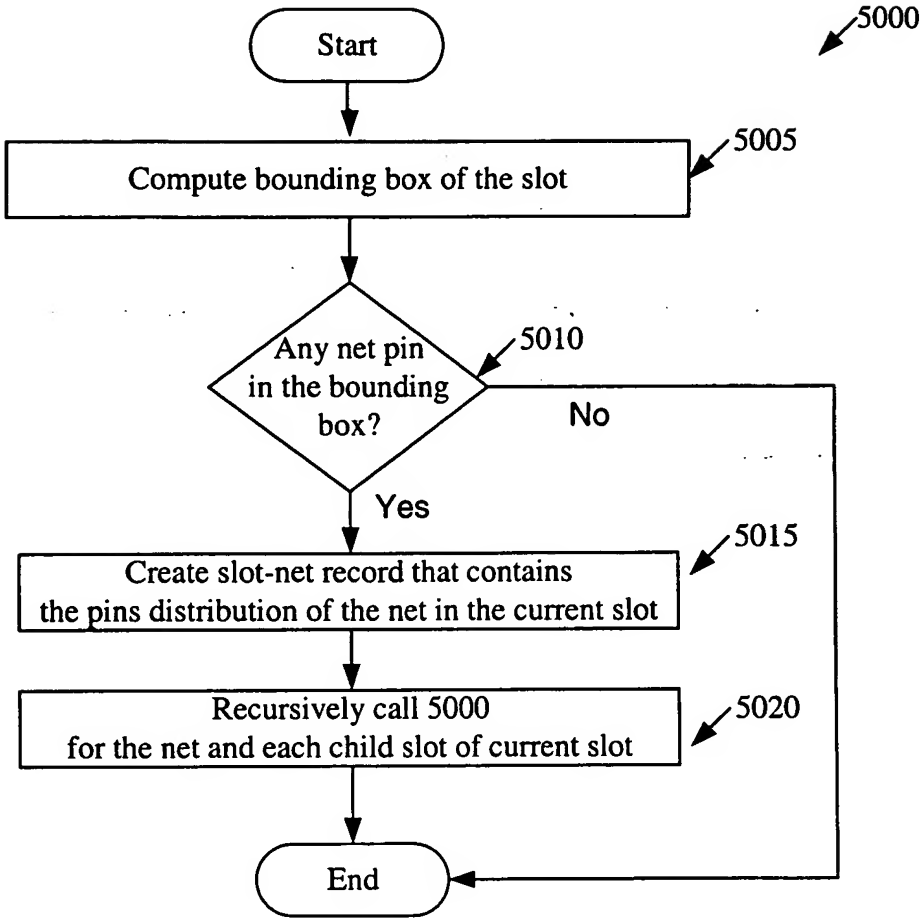
5000
Start
Compute bounding box of the slot
5005
Any net pin in the bounding box?
5010
No
Yes
Create slot-net record that contains the pins distribution of the net in the current slot
5015
Recursively call 5000 for the net and each child slot of current slot
5020
End

Figure 50

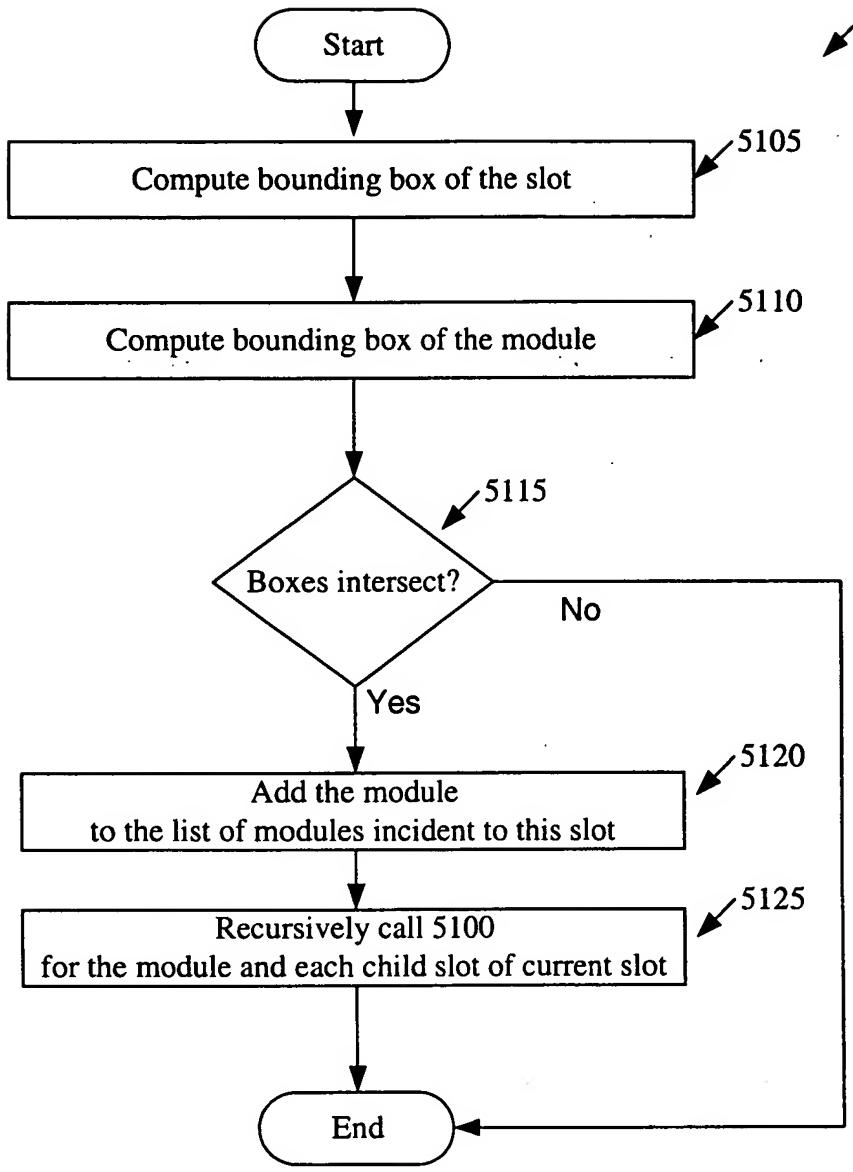
5100
Start
Compute bounding box of the slot
5105
Compute bounding box of the module
5110
5115
Boxes intersect?
No
Yes
Add the module
to the list of modules incident to this slot
5120
Recursively call 5100
for the module and each child slot of current slot
5125
End

Figure 51

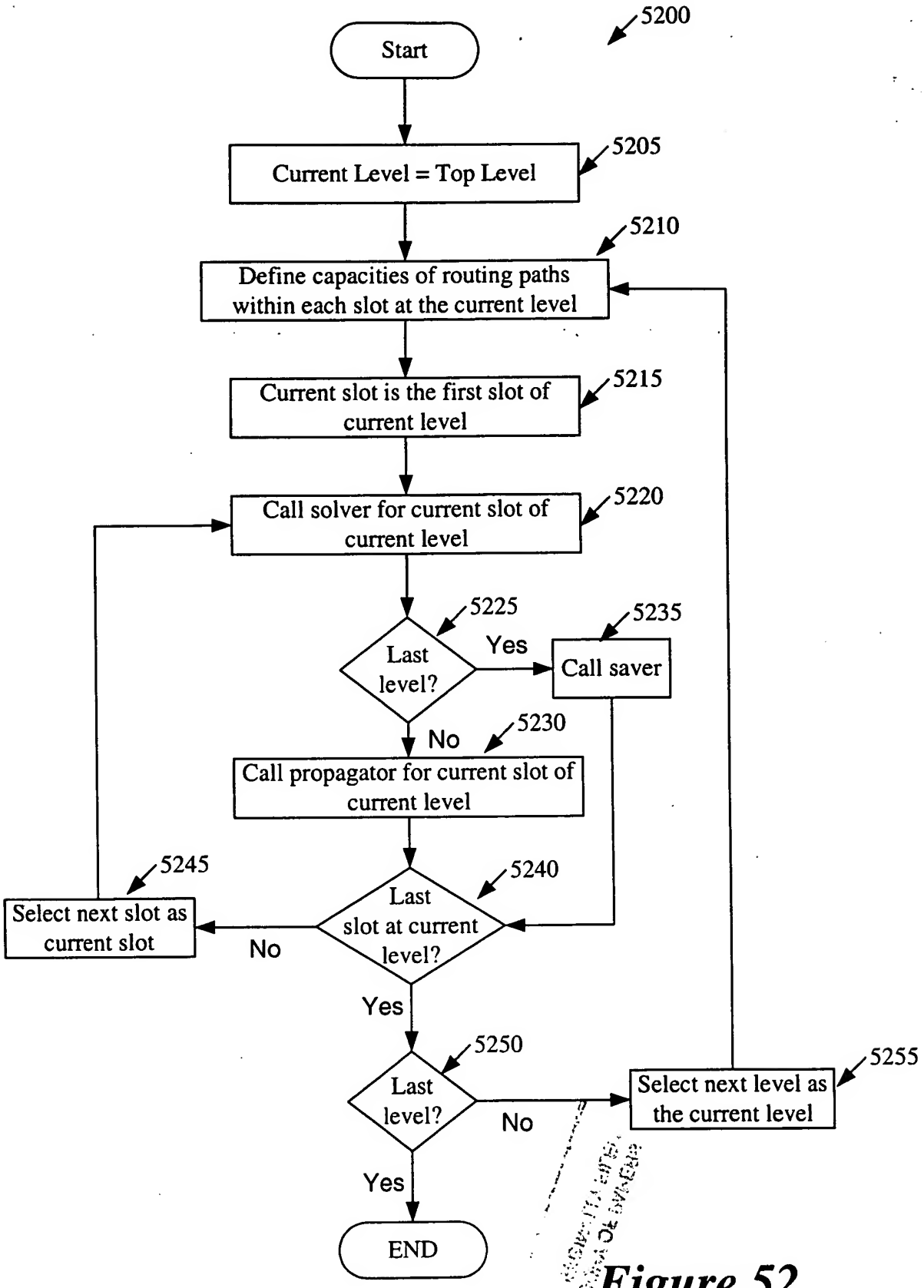
5200
Start
5205
Current Level = Top Level
5210
Define capacities of routing paths within each slot at the current level
5215
Current slot is the first slot of current level
5220
Call solver for current slot of current level
5225
Last level?
Yes
5235
Call saver
No
5230
Call propagator for current slot of current level
5245
Select next slot as current slot
5240
Last slot at current level?
No
Yes
5250
Last level?
No
5255
Select next level as the current level
Yes
END

Figure 52

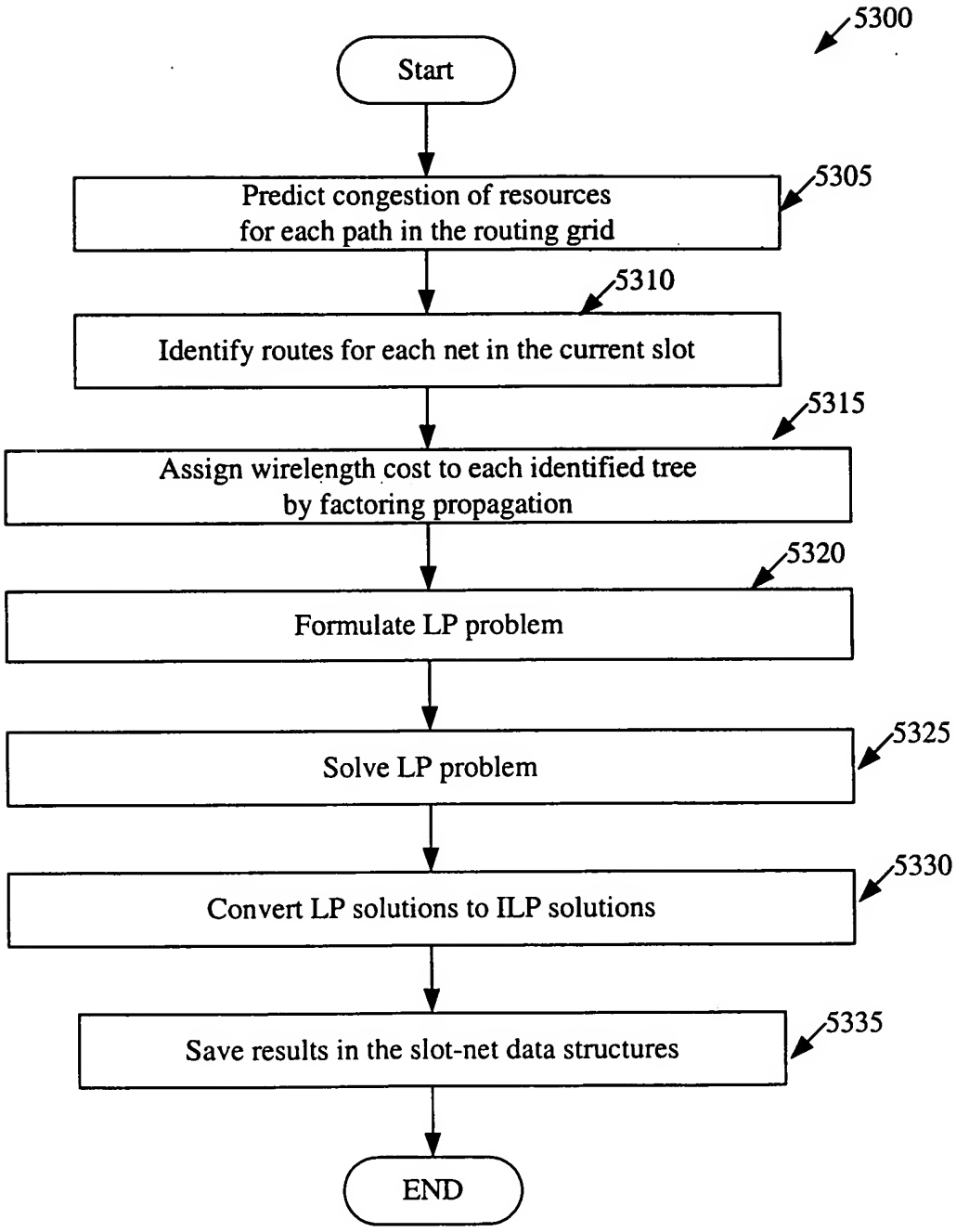
5300
Start
Predict congestion of resources for each path in the routing grid
5305
5310
Identify routes for each net in the current slot
5315
Assign wirelength cost to each identified tree by factoring propagation
5320
Formulate LP problem
5325
Solve LP problem
5330
Convert LP solutions to ILP solutions
5335
Save results in the slot-net data structures
END

Figure 53

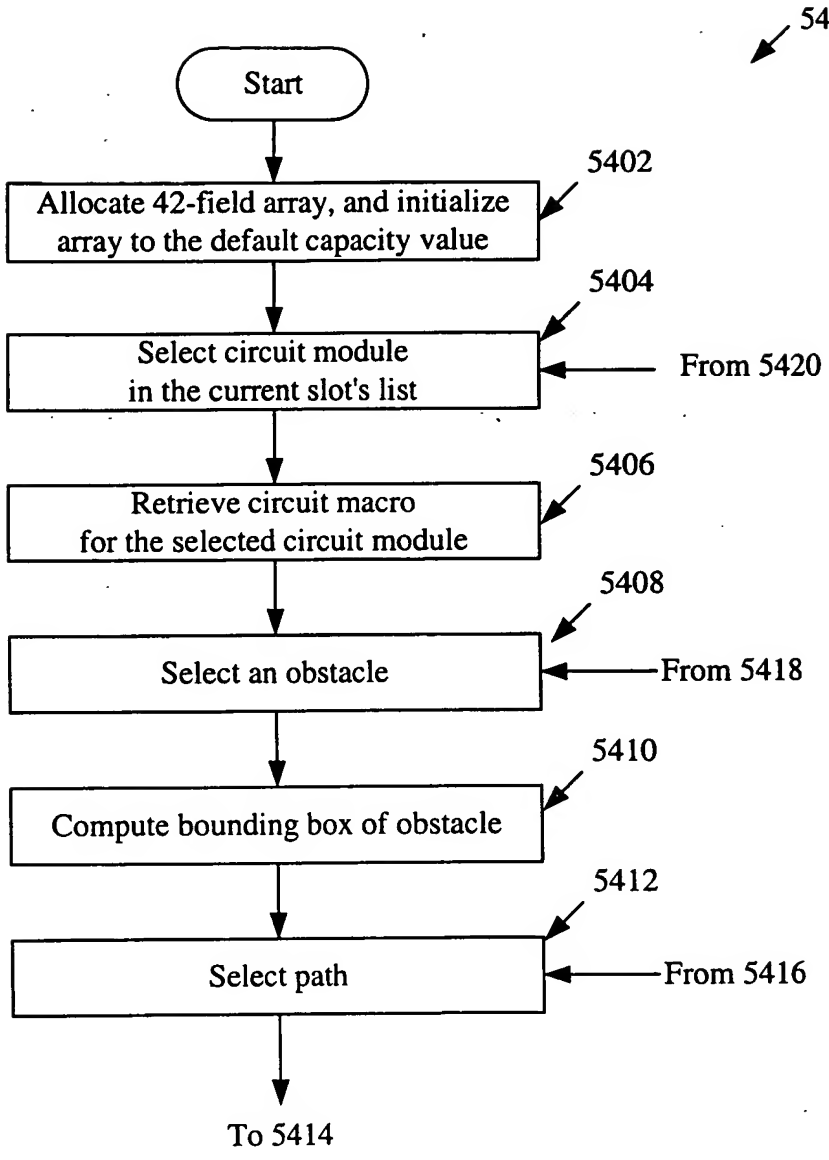

Figure 54A

Figure 54: $\frac{\textit{Figure 54A}}{\textit{Figure 54B}}$

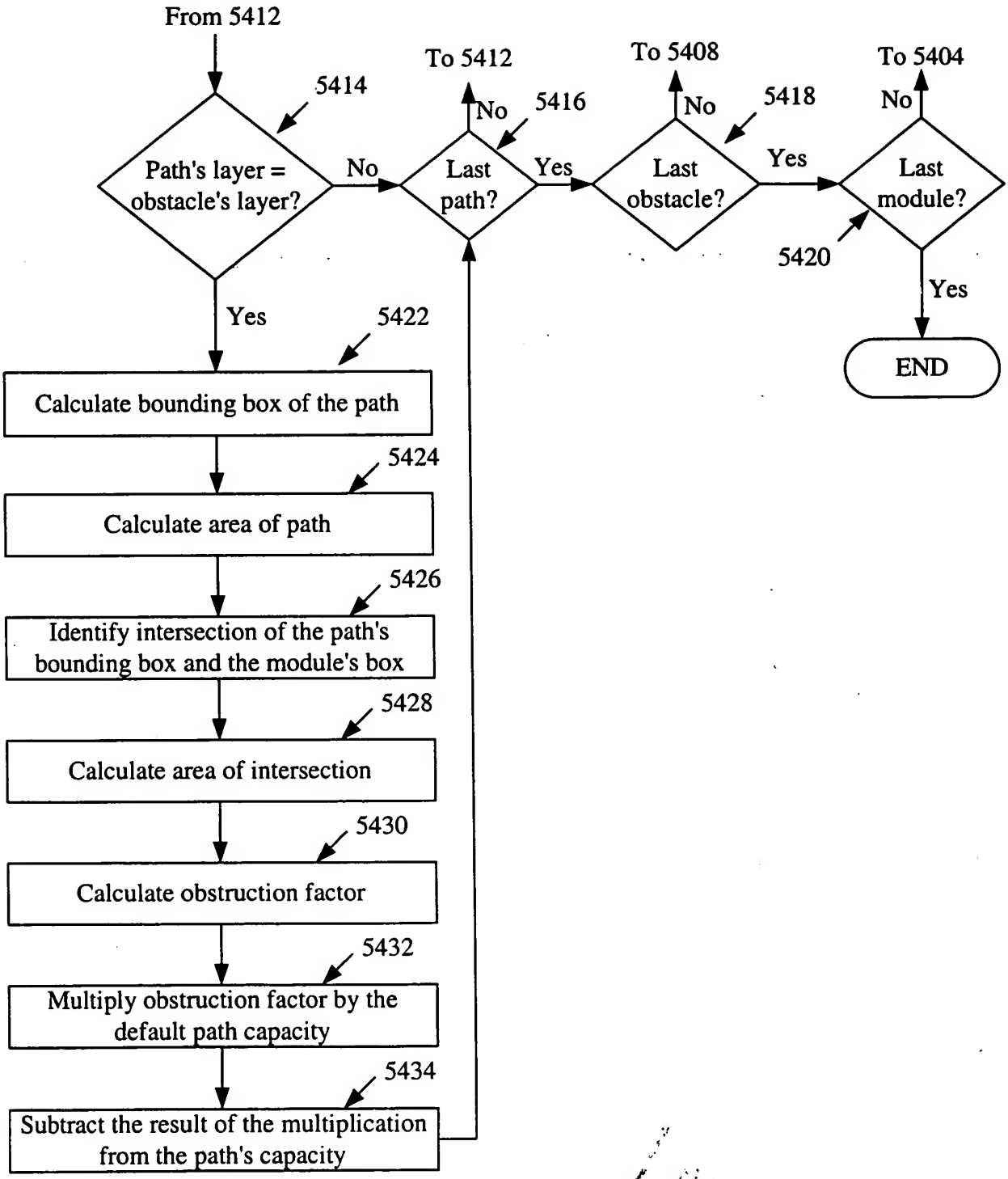
From 5412
5414
Path's layer = obstacle's layer?
No
To 5412
No
5416
Last path?
Yes
To 5408
No
5418
Last obstacle?
Yes
To 5404
No
Last module?
5420
Yes
END
Yes
5422
Calculate bounding box of the path
5424
Calculate area of path
5426
Identify intersection of the path's bounding box and the module's box
5428
Calculate area of intersection
5430
Calculate obstruction factor
5432
Multiply obstruction factor by the default path capacity
5434
Subtract the result of the multiplication from the path's capacity

Figure 54B

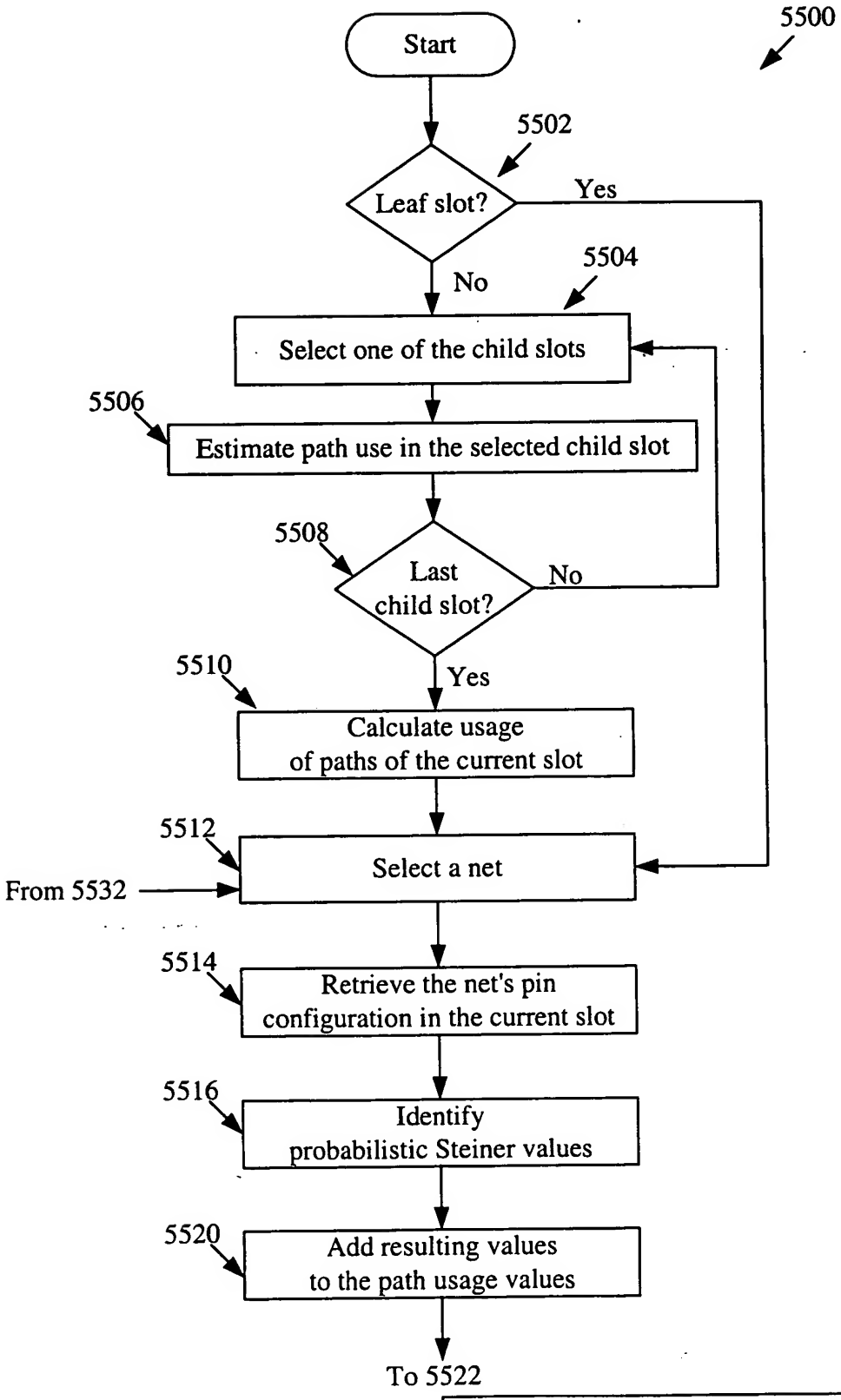

Figure 55A

Figure 55A

Figure 55: Figure 55B

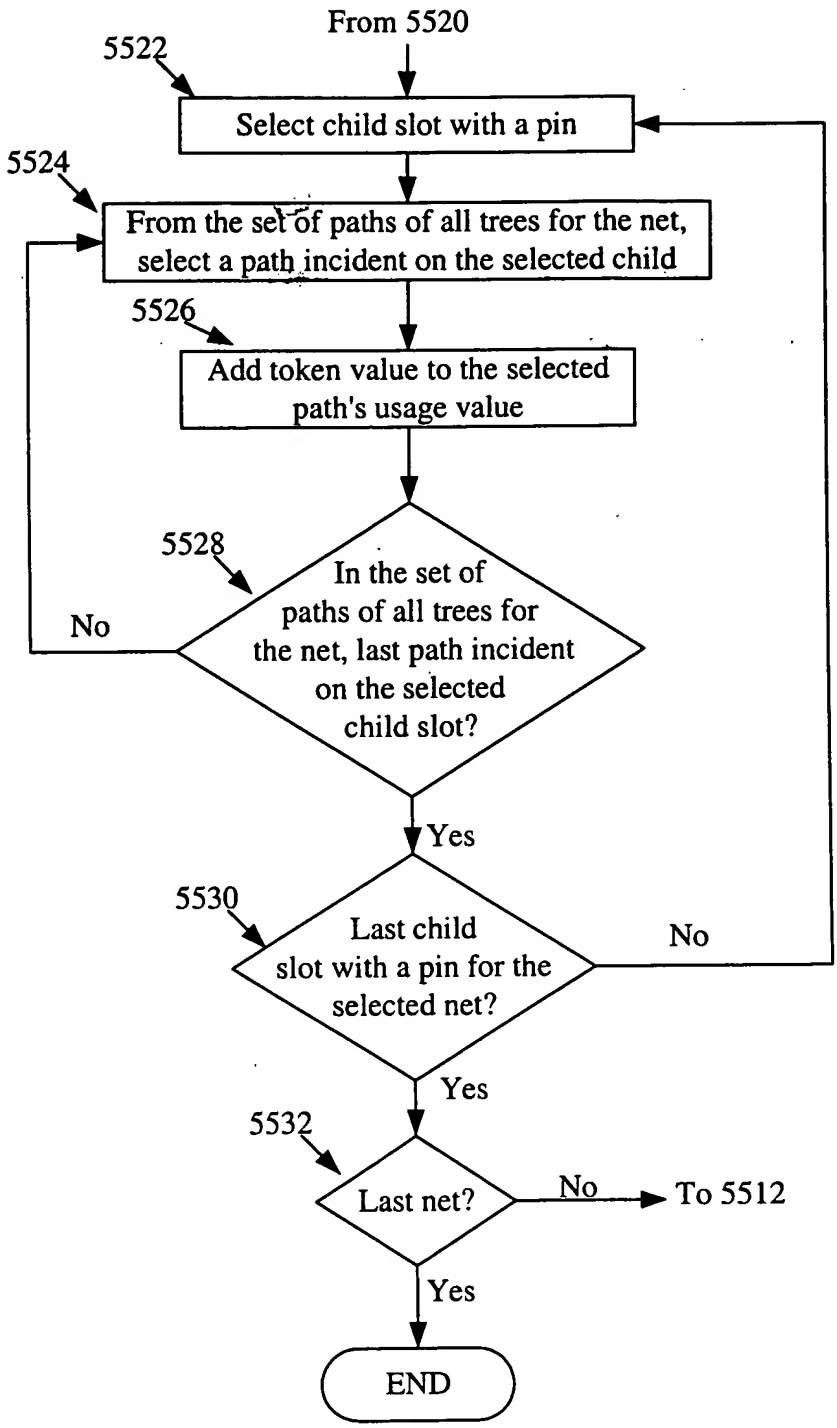
From 5520
5522
Select child slot with a pin
5524
From the set of paths of all trees for the net, select a path incident on the selected child
5526
Add token value to the selected path's usage value
5528
In the set of paths of all trees for the net, last path incident on the selected child slot?
No
Yes
5530
Last child slot with a pin for the selected net?
No
Yes
5532
Last net?
No
To 5512
Yes
END

Figure 55B

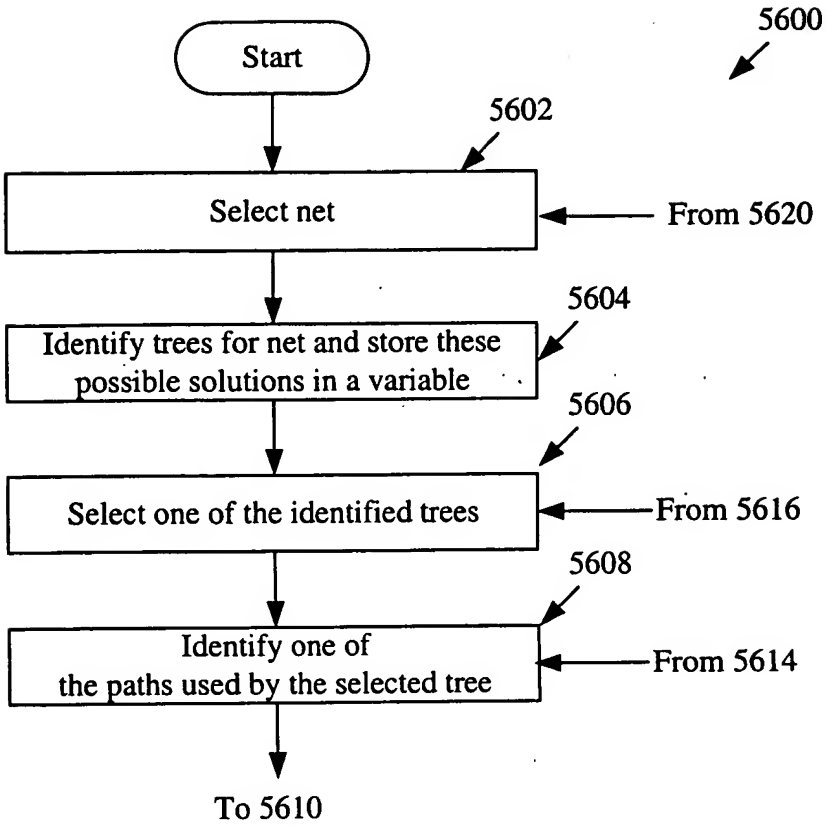

Figure 56A

Figure 56: *Figure 56A* / *Figure 56B* / *Figure 56C* / *Figure 56D*

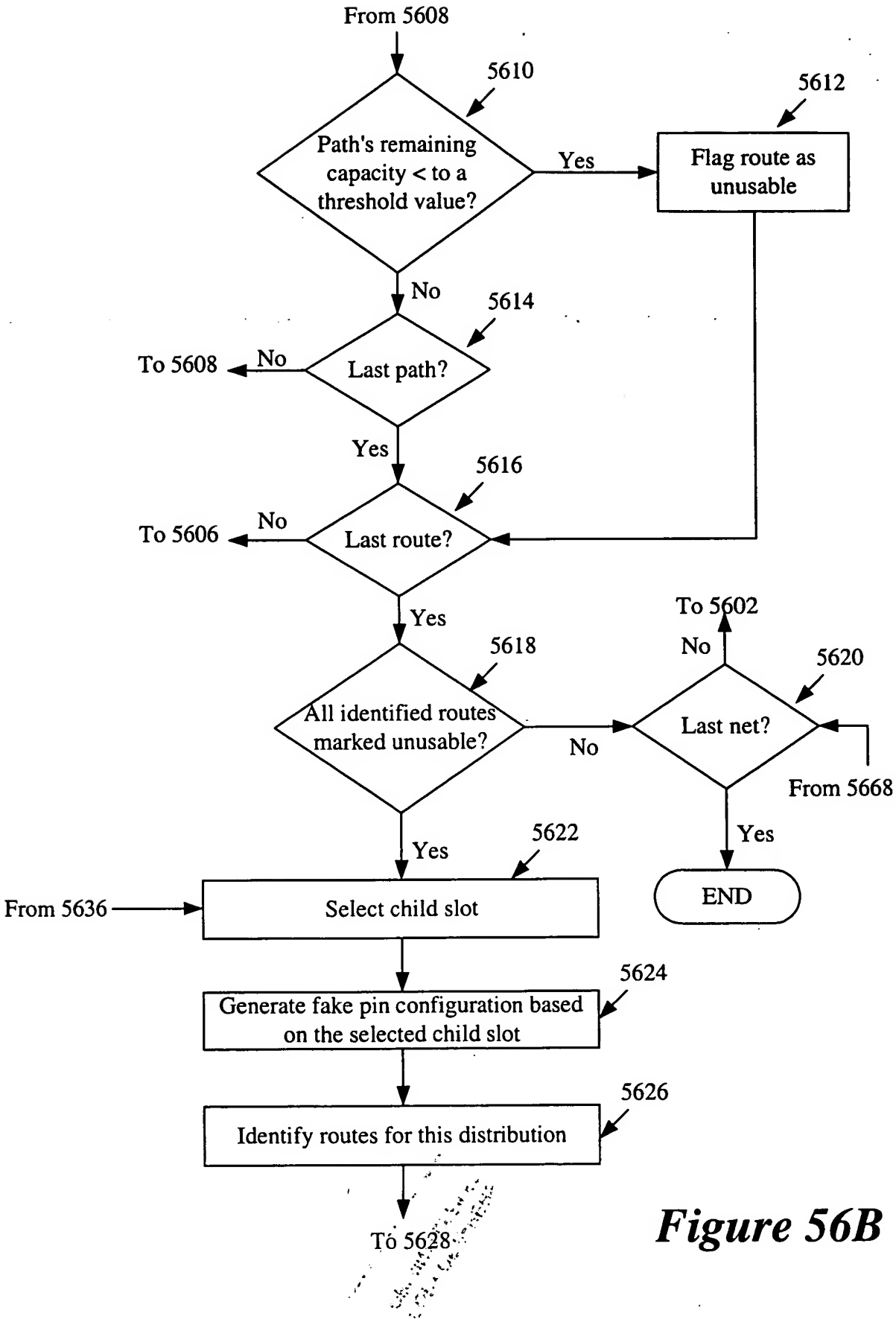
From 5608
5610
Path's remaining capacity < to a threshold value?
Yes
5612
Flag route as unusable
No
5614
Last path?
No
To 5608
Yes
5616
Last route?
No
To 5606
Yes
5618
All identified routes marked unusable?
No
To 5602
No
5620
Last net?
From 5668
Yes
END
Yes
5622
From 5636
Select child slot
5624
Generate fake pin configuration based on the selected child slot
5626
Identify routes for this distribution
To 5628

Figure 56B

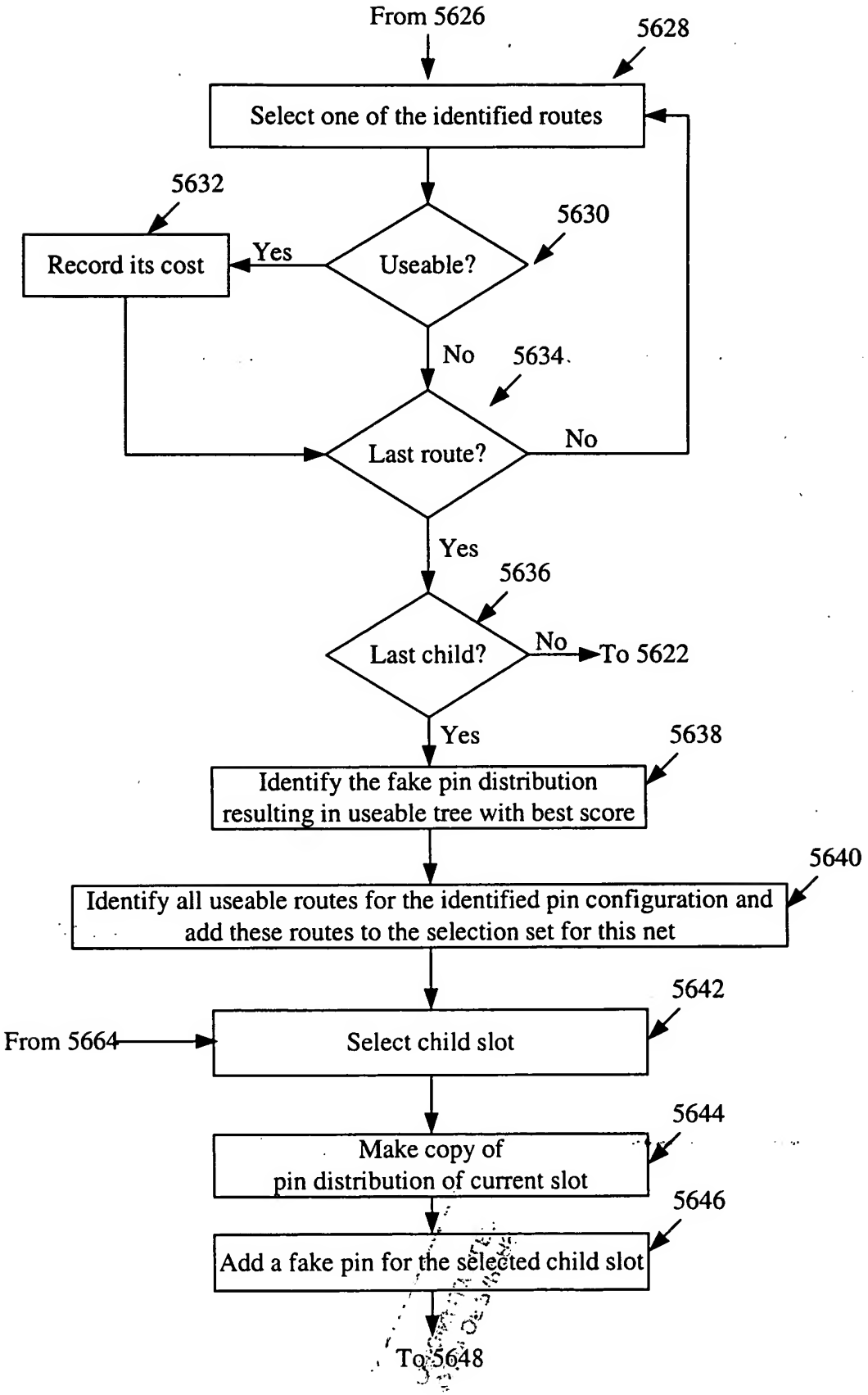
From 5626
5628
Select one of the identified routes
5632
5630
Record its cost
Yes
Useable?
No
5634
Last route?
No
Yes
5636
Last child?
No
To 5622
Yes
5638
Identify the fake pin distribution resulting in useable tree with best score
5640
Identify all useable routes for the identified pin configuration and add these routes to the selection set for this net
5642
From 5664
Select child slot
5644
Make copy of pin distribution of current slot
5646
Add a fake pin for the selected child slot
To 5648

Figure 56C

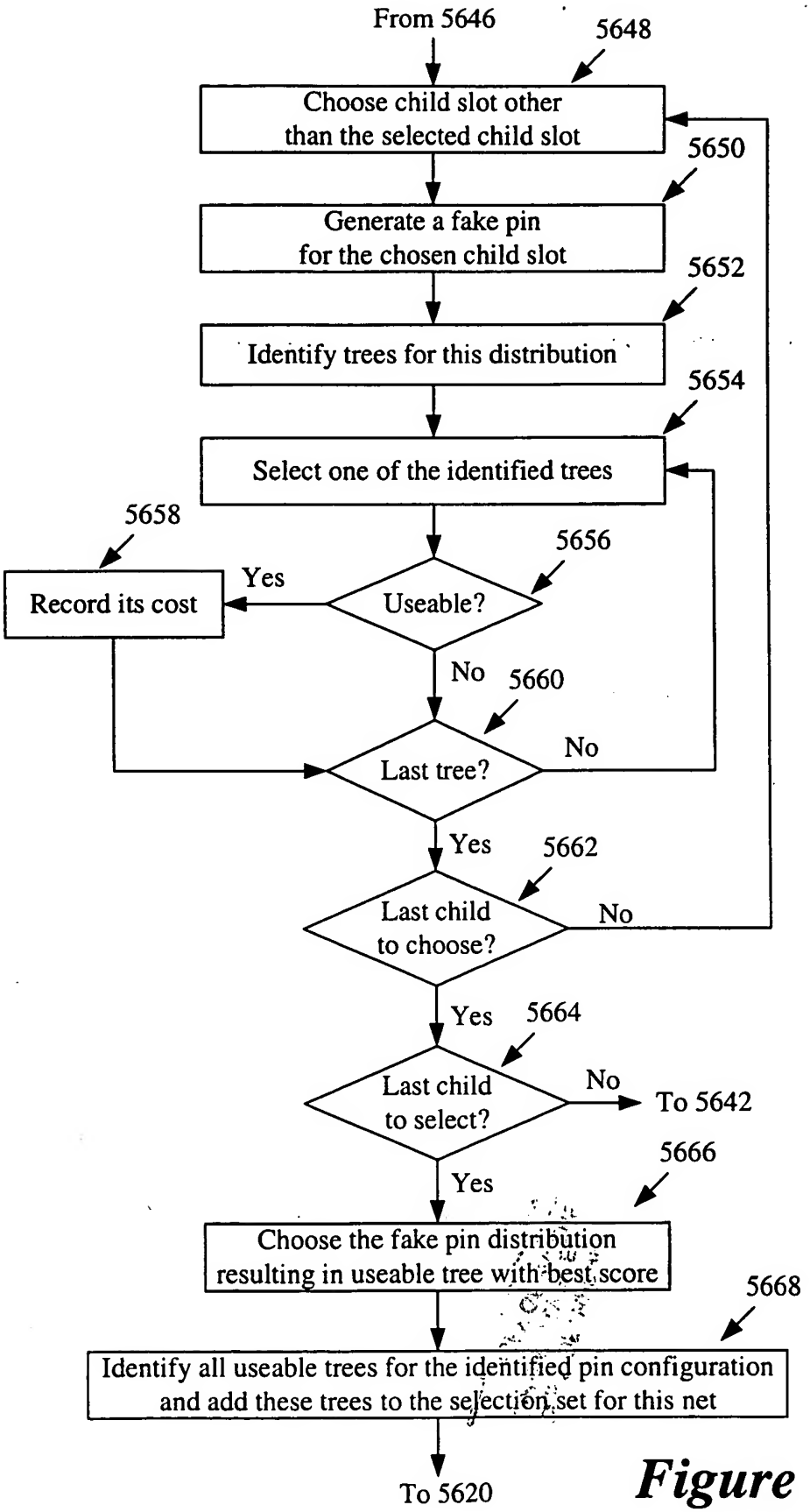
From 5646
5648
Choose child slot other than the selected child slot
5650
Generate a fake pin for the chosen child slot
5652
Identify trees for this distribution
5654
Select one of the identified trees
5658
5656
Record its cost
Yes
Useable?
No
5660
Last tree?
No
Yes
5662
Last child to choose?
No
Yes
5664
Last child to select?
No
To 5642
5666
Yes
Choose the fake pin distribution resulting in useable tree with best score
5668
Identify all useable trees for the identified pin configuration and add these trees to the selection set for this net
To 5620

Figure 56D

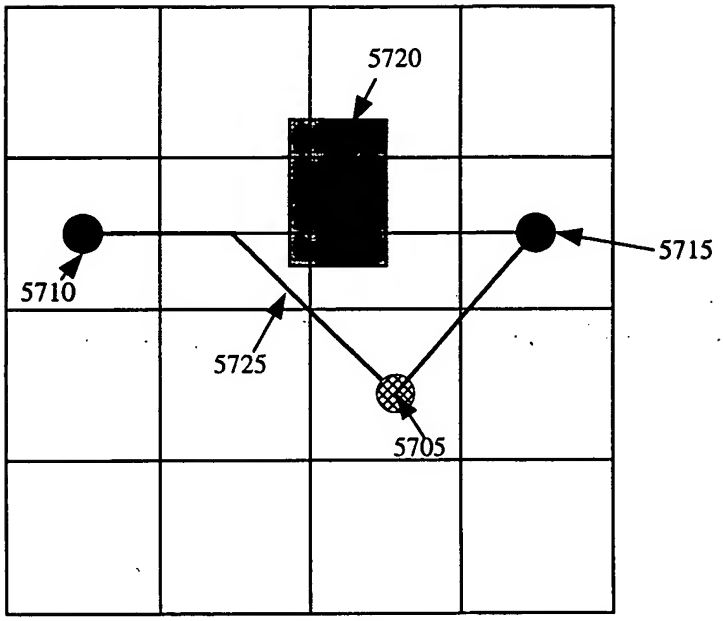

Figure 57

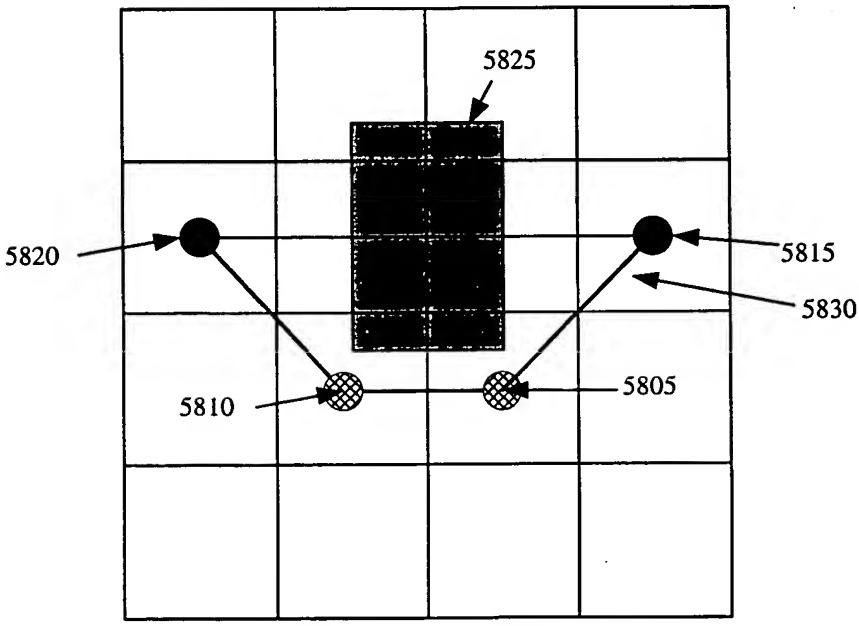

Figure 58

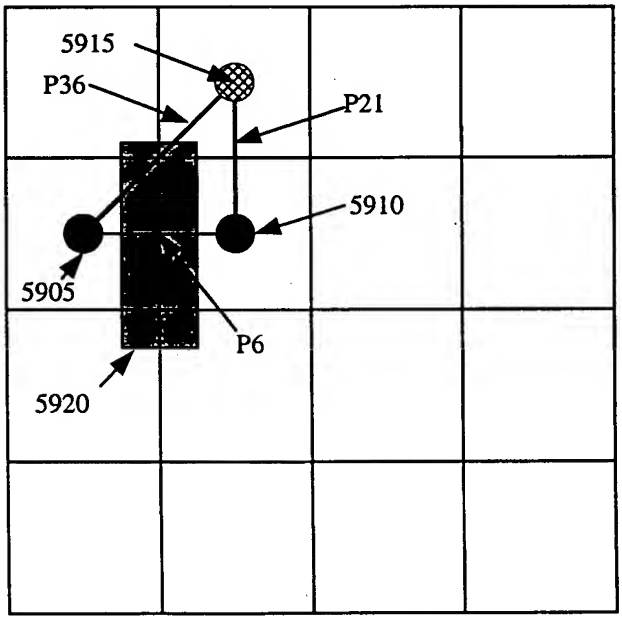

Figure 59

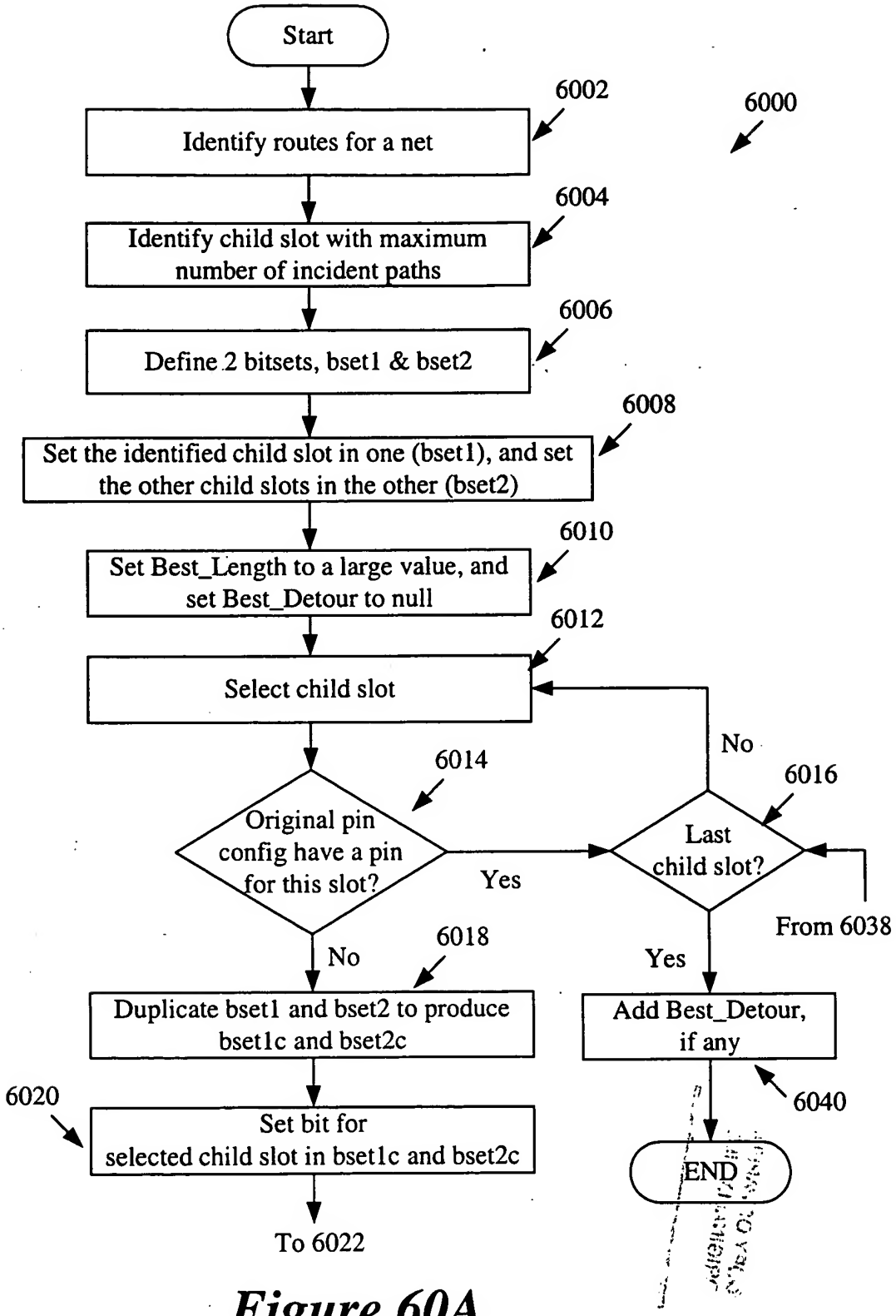

Figure 60A

Figure 60: Figure 60A / Figure 60B

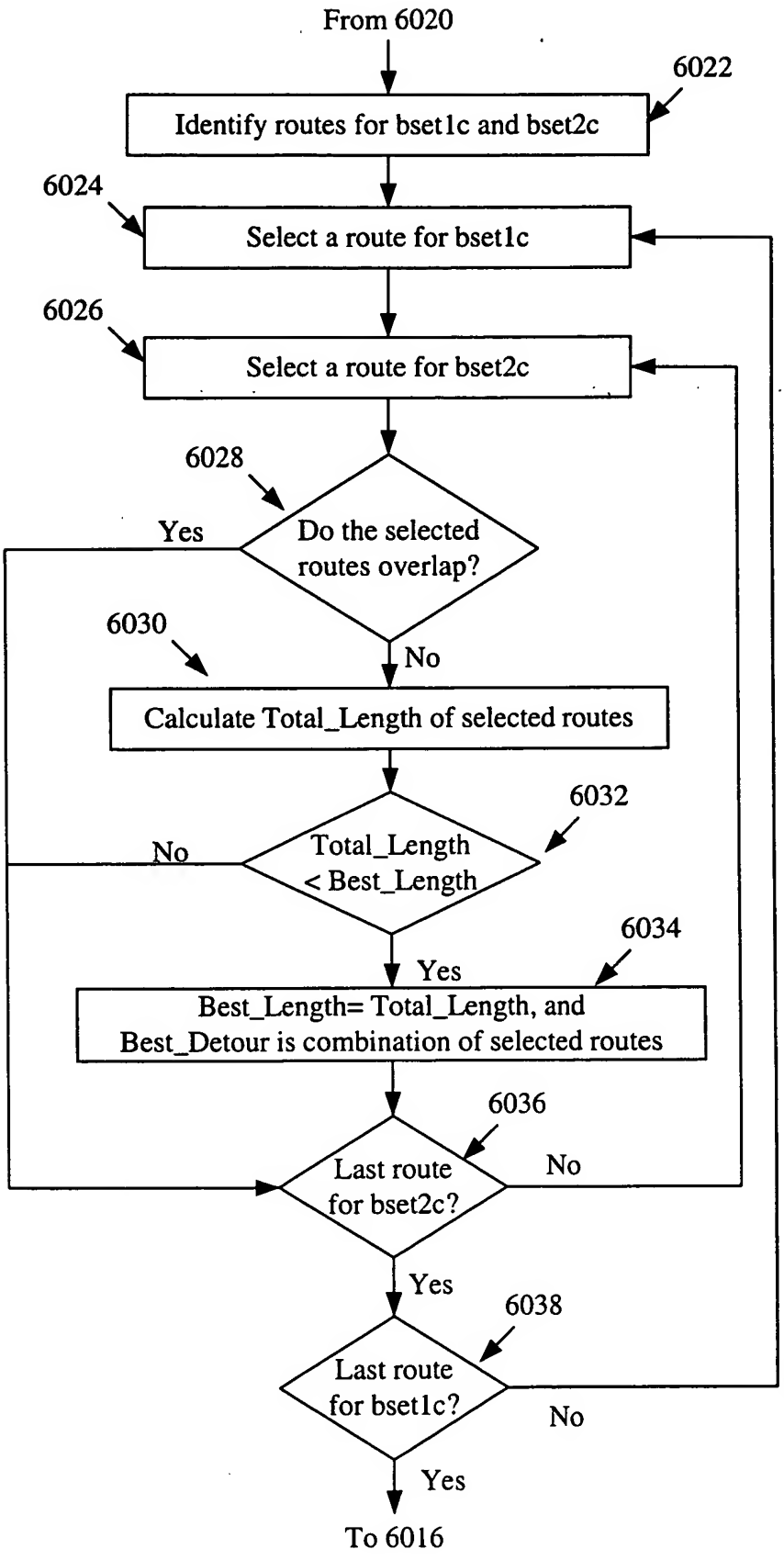
From 6020
6022
Identify routes for bset1c and bset2c
6024
Select a route for bset1c
6026
Select a route for bset2c
6028
Do the selected routes overlap?
Yes
No
6030
Calculate Total_Length of selected routes
6032
Total_Length < Best_Length
No
Yes
6034
Best_Length= Total_Length, and Best_Detour is combination of selected routes
6036
Last route for bset2c?
No
Yes
6038
Last route for bset1c?
No
Yes
To 6016

Figure 60B

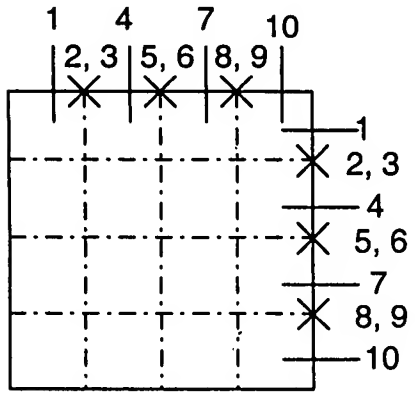

Figure 61

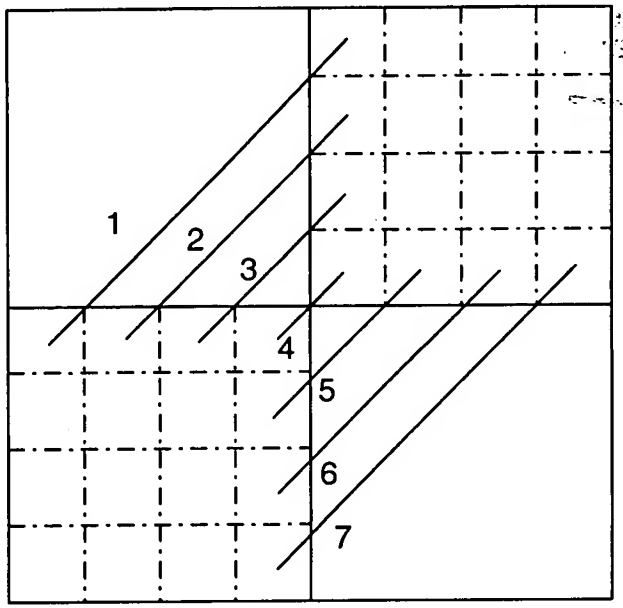

Figure 62

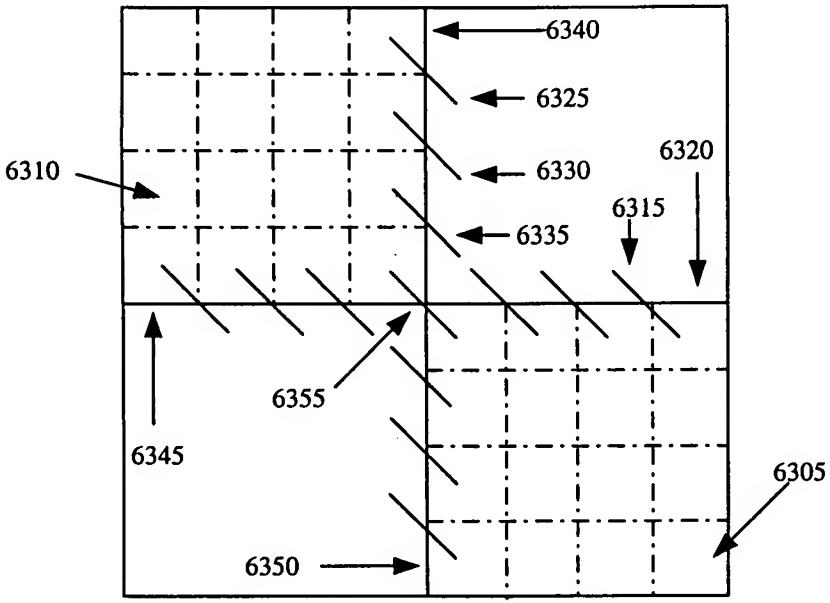

Figure 63

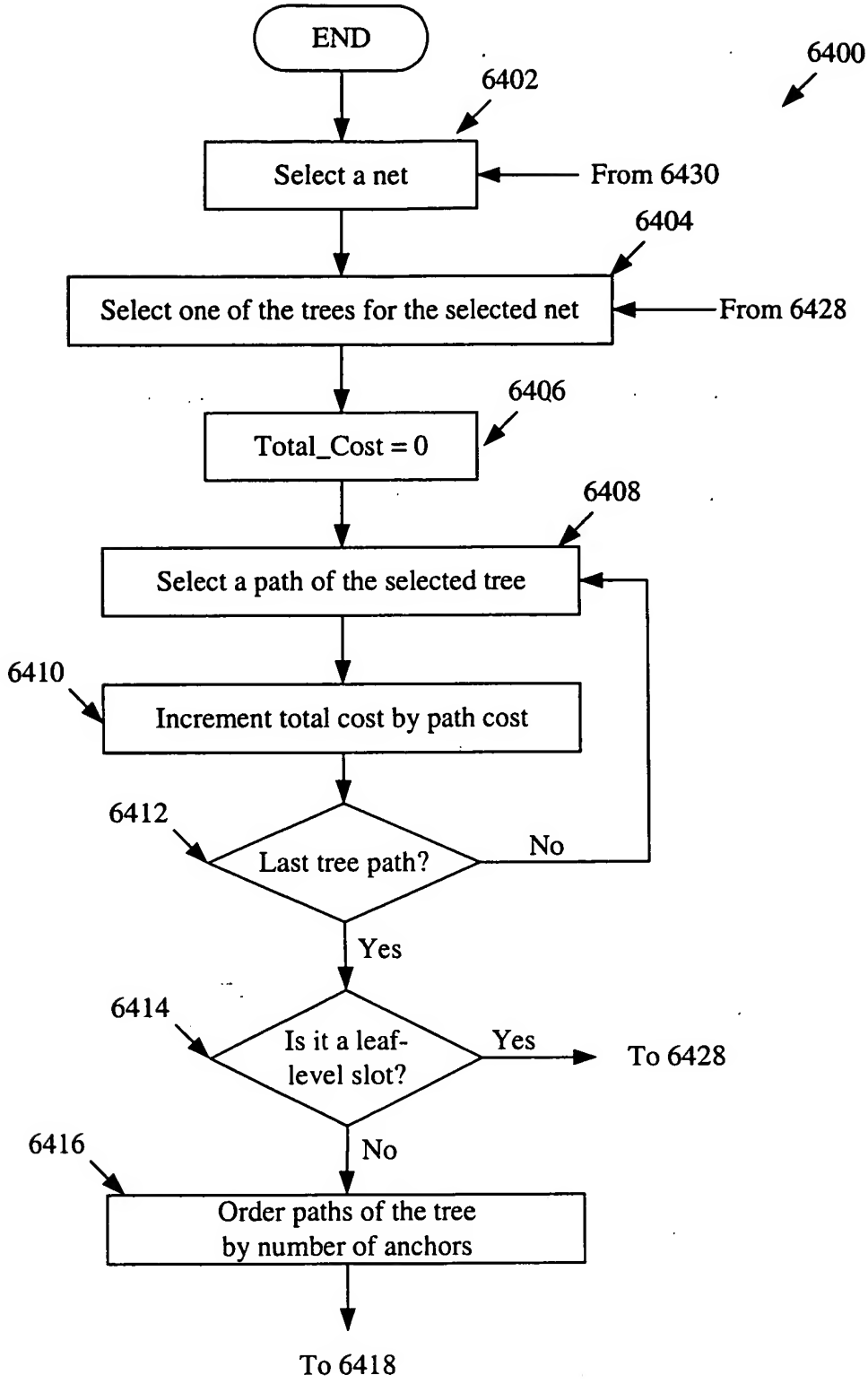

Figure 64A

Figure 64: Figure 64A / Figure 64B

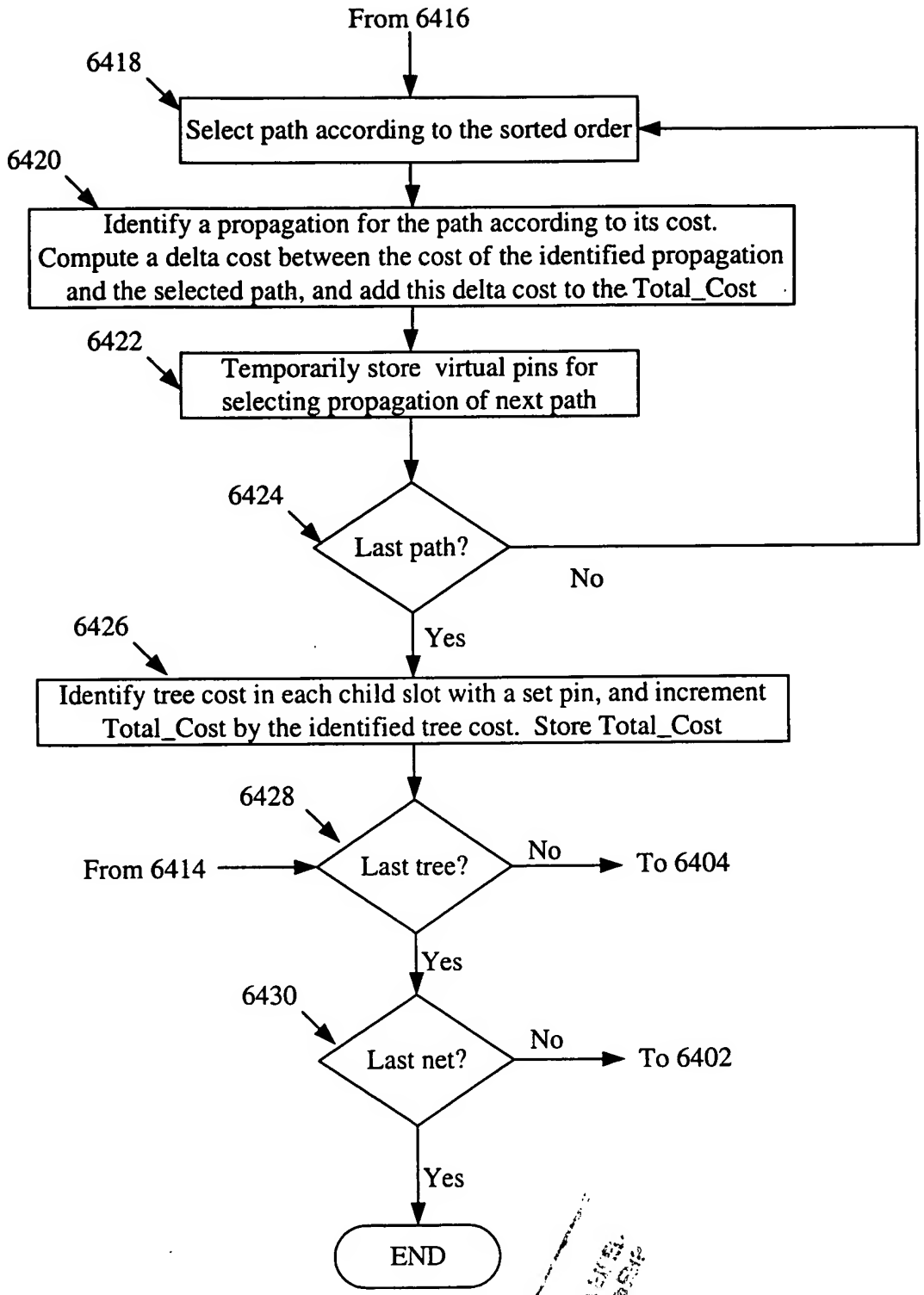
From 6416
6418
Select path according to the sorted order
6420
Identify a propagation for the path according to its cost. Compute a delta cost between the cost of the identified propagation and the selected path, and add this delta cost to the Total_Cost
6422
Temporarily store virtual pins for selecting propagation of next path
6424
Last path?
No
Yes
6426
Identify tree cost in each child slot with a set pin, and increment Total_Cost by the identified tree cost. Store Total_Cost
6428
From 6414
Last tree?
No
To 6404
Yes
6430
Last net?
No
To 6402
Yes
END

Figure 64B

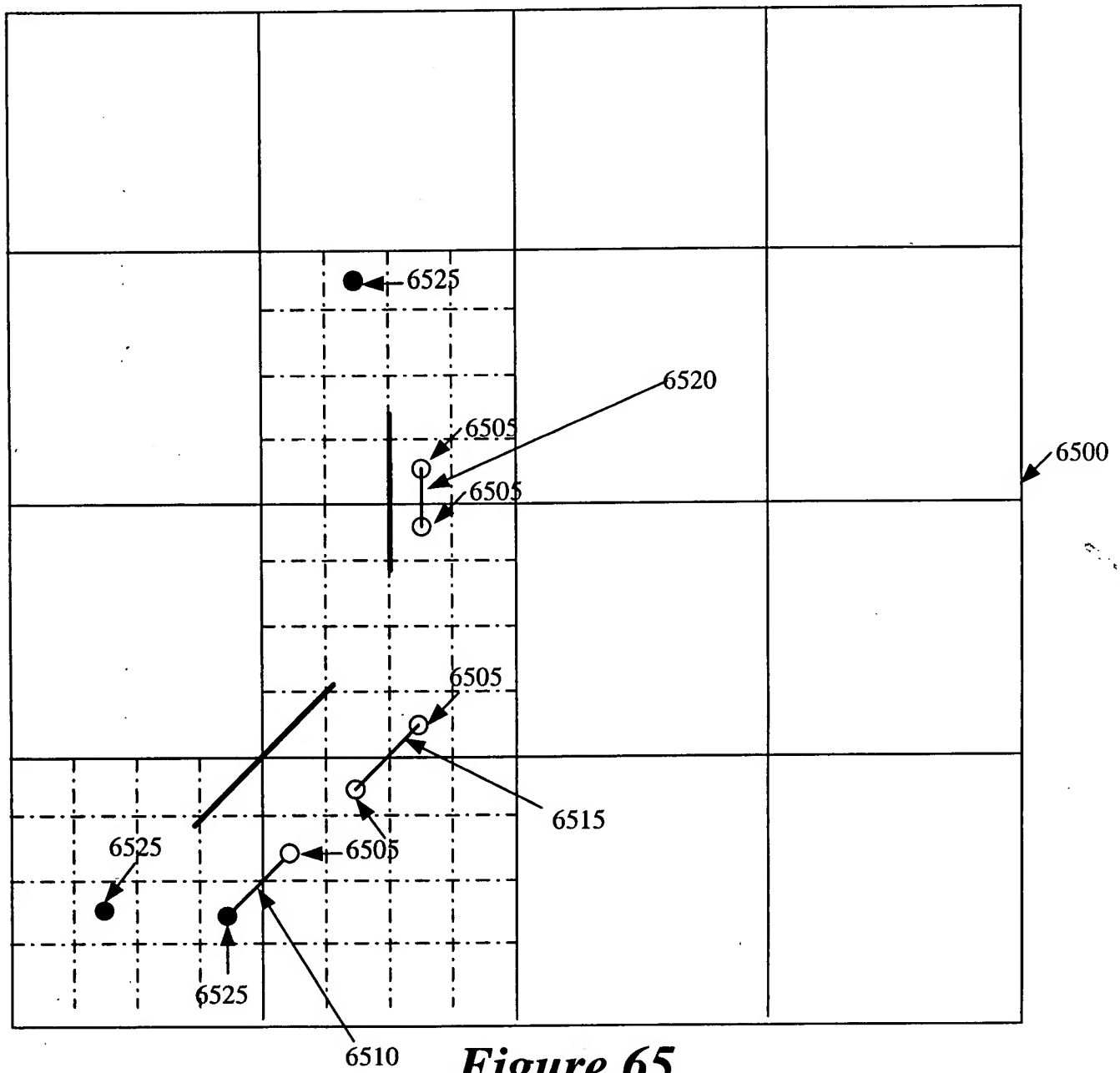

Figure 65

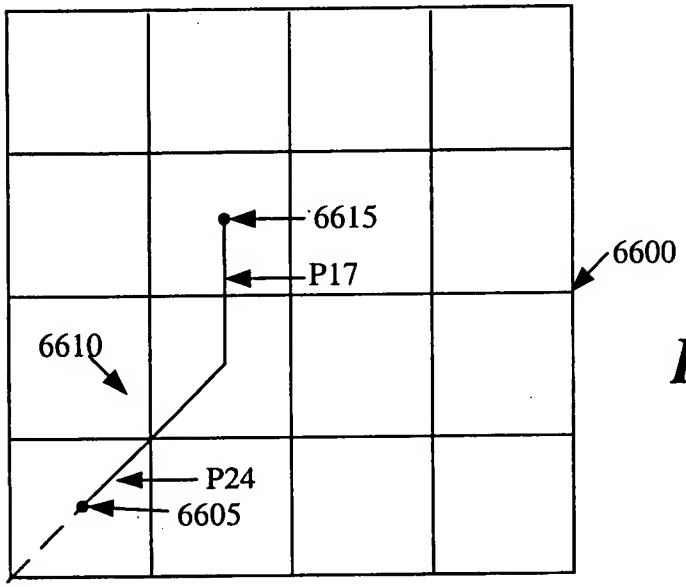

Figure 66

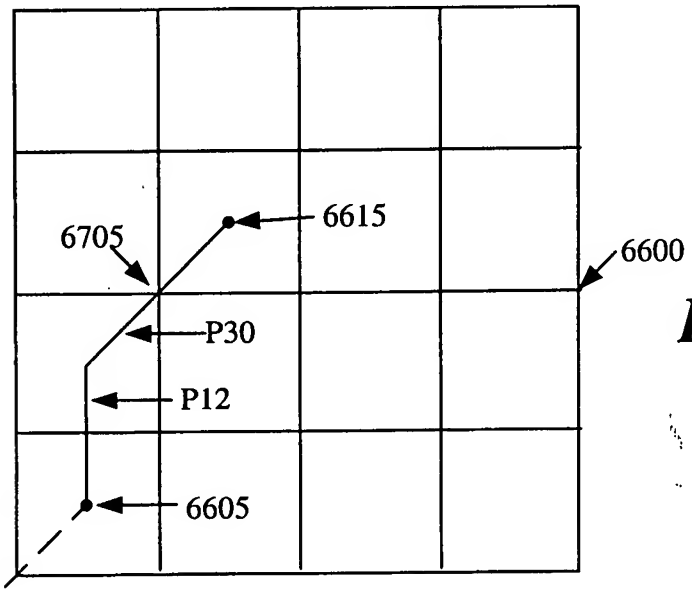

Figure 67

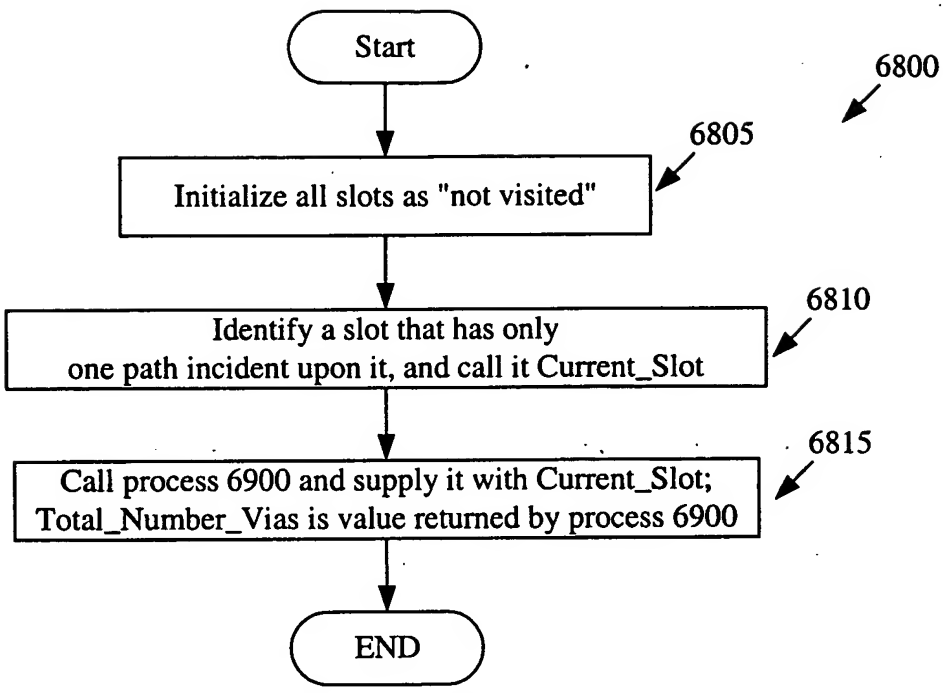

Figure 68

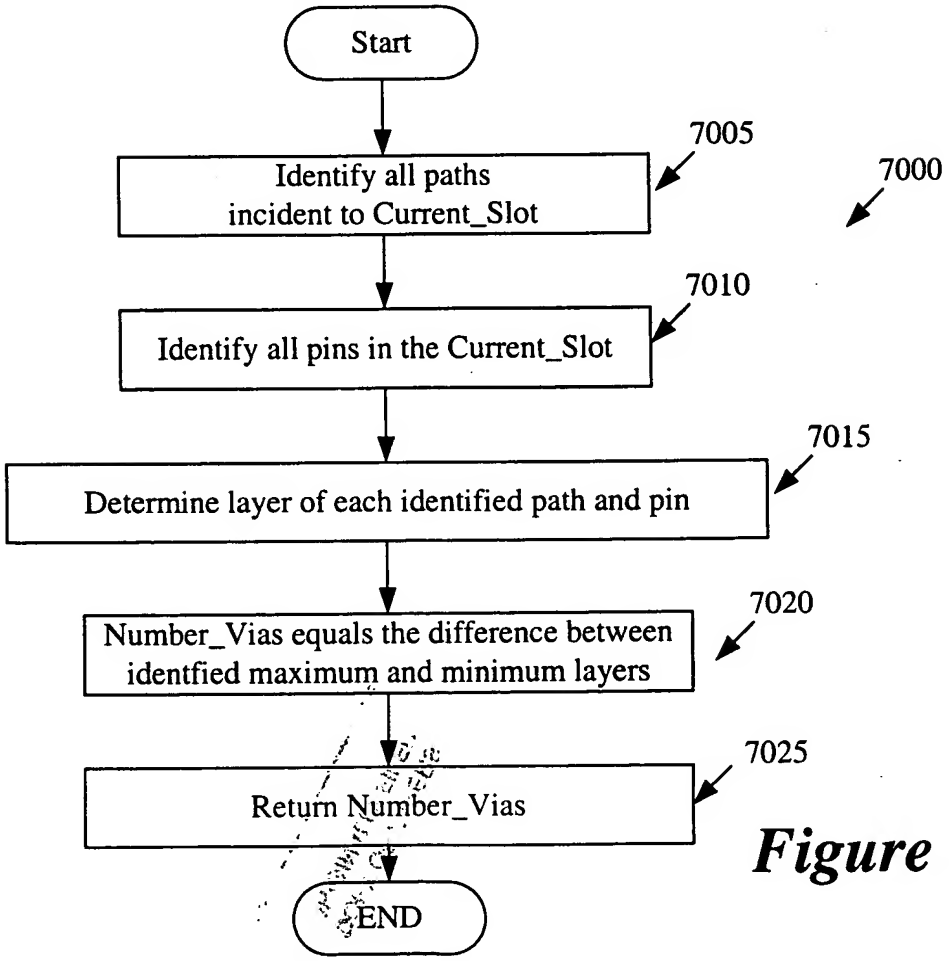

Figure 70

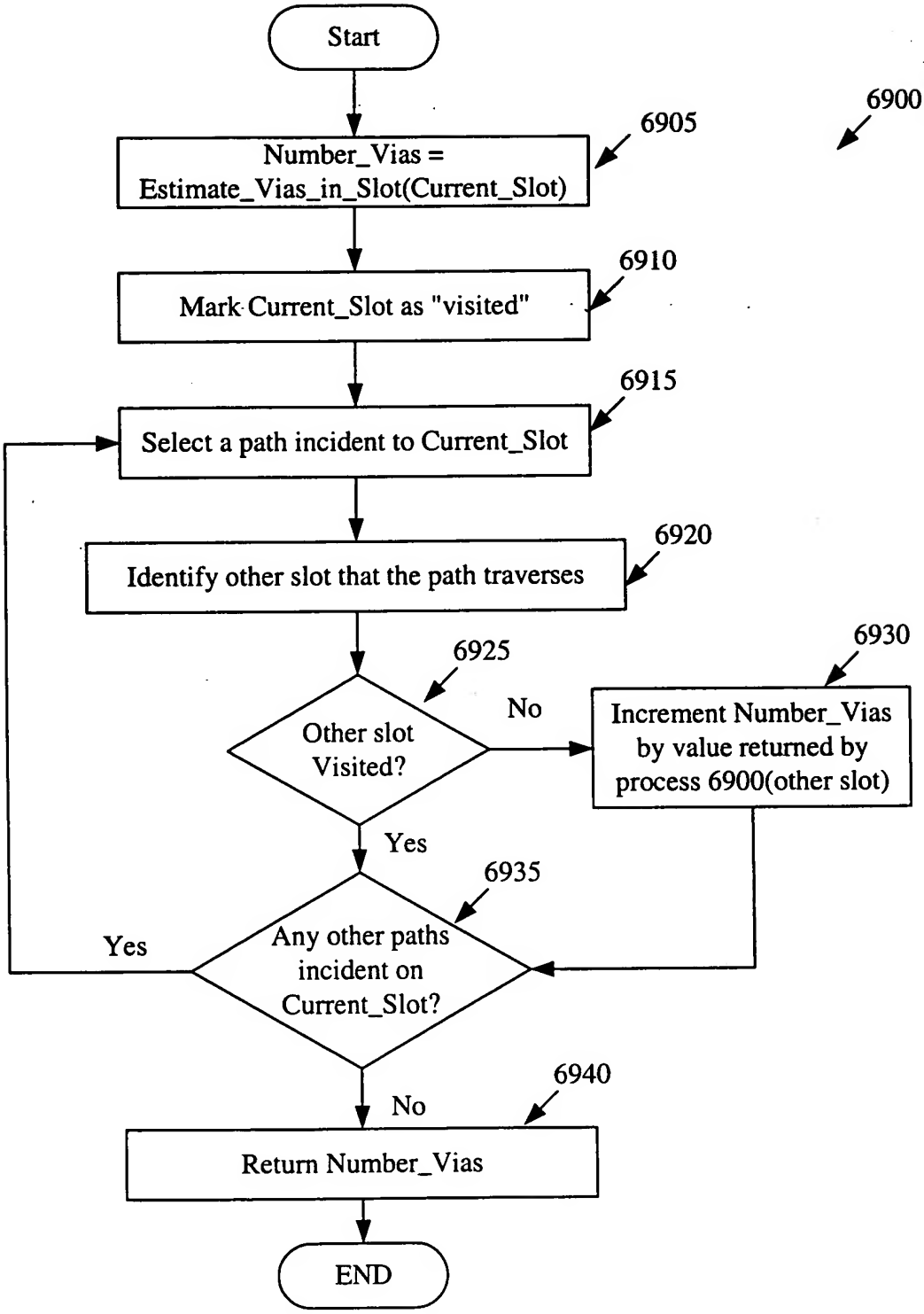
Start
6900
6905
Number_Vias = Estimate_Vias_in_Slot(Current_Slot)
6910
Mark Current_Slot as "visited"
6915
Select a path incident to Current_Slot
6920
Identify other slot that the path traverses
6925
Other slot Visited?
No
6930
Increment Number_Vias by value returned by process 6900(other slot)
Yes
6935
Any other paths incident on Current_Slot?
Yes
No
6940
Return Number_Vias
END

Figure 69

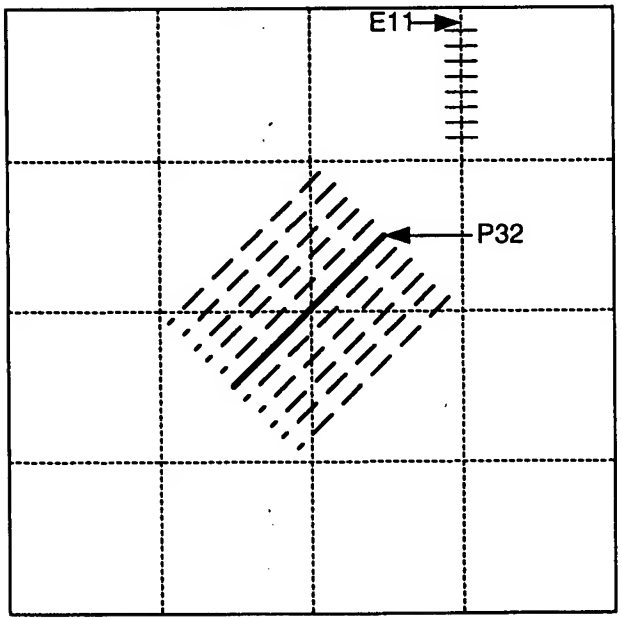

Figure 71

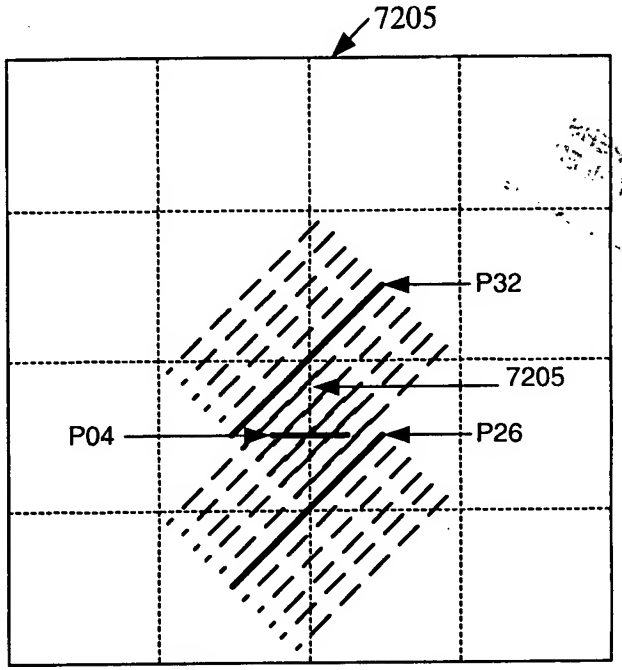

Figure 72

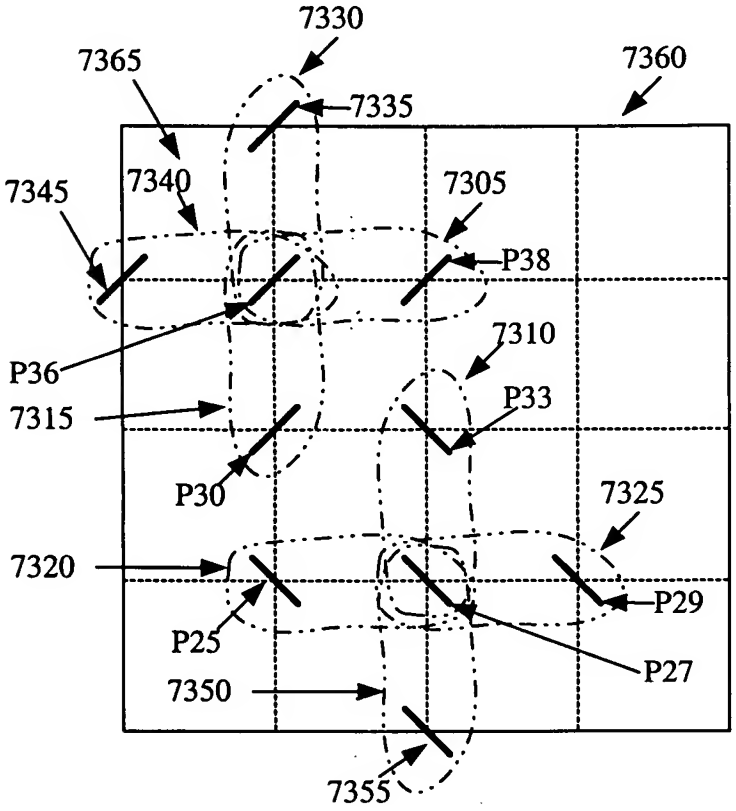

Figure 73

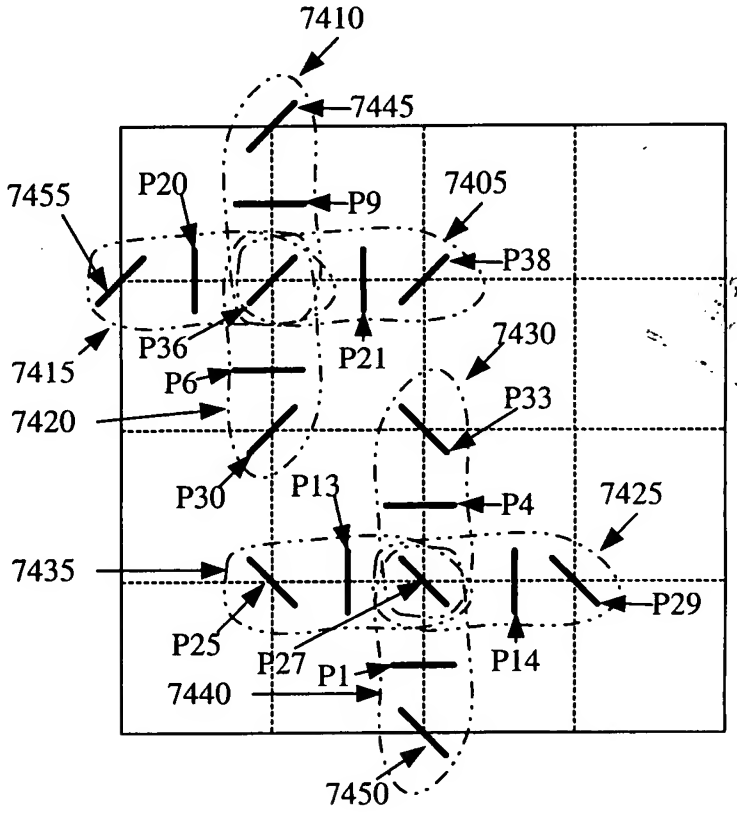

Figure 74

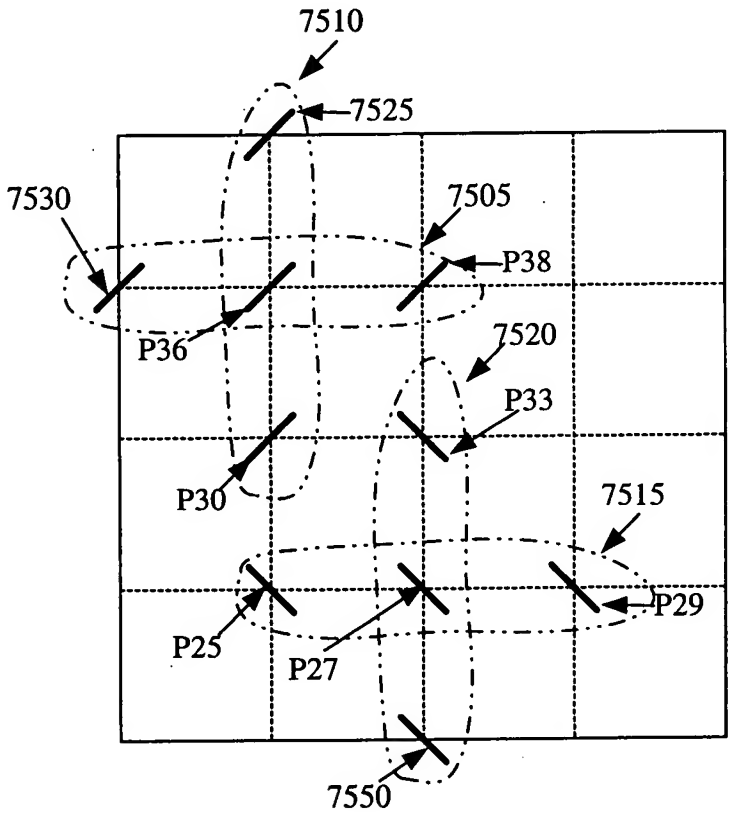

Figure 75

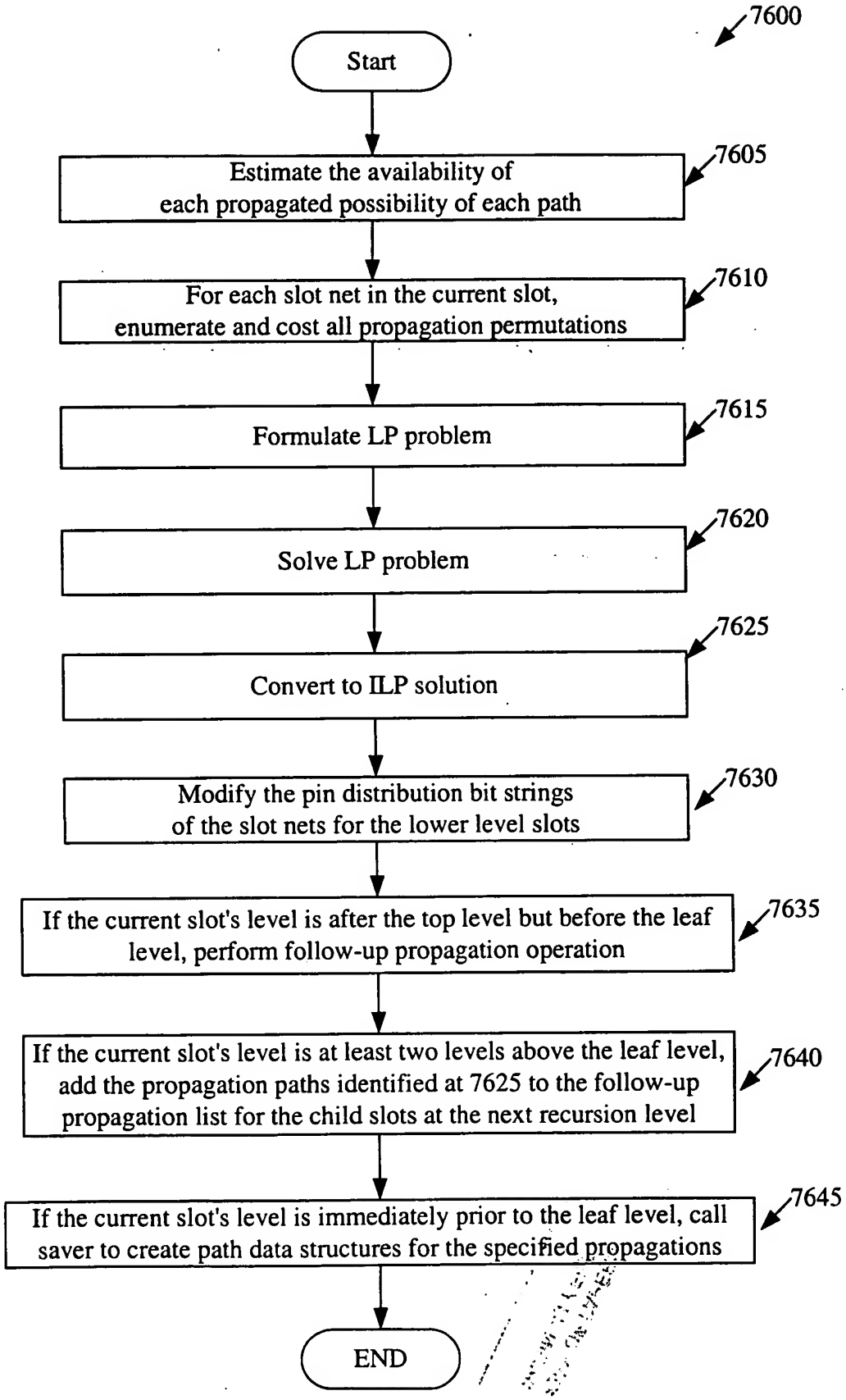
7600
Start
Estimate the availability of each propagated possibility of each path
7605
For each slot net in the current slot, enumerate and cost all propagation permutations
7610
Formulate LP problem
7615
Solve LP problem
7620
Convert to ILP solution
7625
Modify the pin distribution bit strings of the slot nets for the lower level slots
7630
If the current slot's level is after the top level but before the leaf level, perform follow-up propagation operation
7635
If the current slot's level is at least two levels above the leaf level, add the propagation paths identified at 7625 to the follow-up propagation list for the child slots at the next recursion level
7640
If the current slot's level is immediately prior to the leaf level, call saver to create path data structures for the specified propagations
7645
END

Figure 76

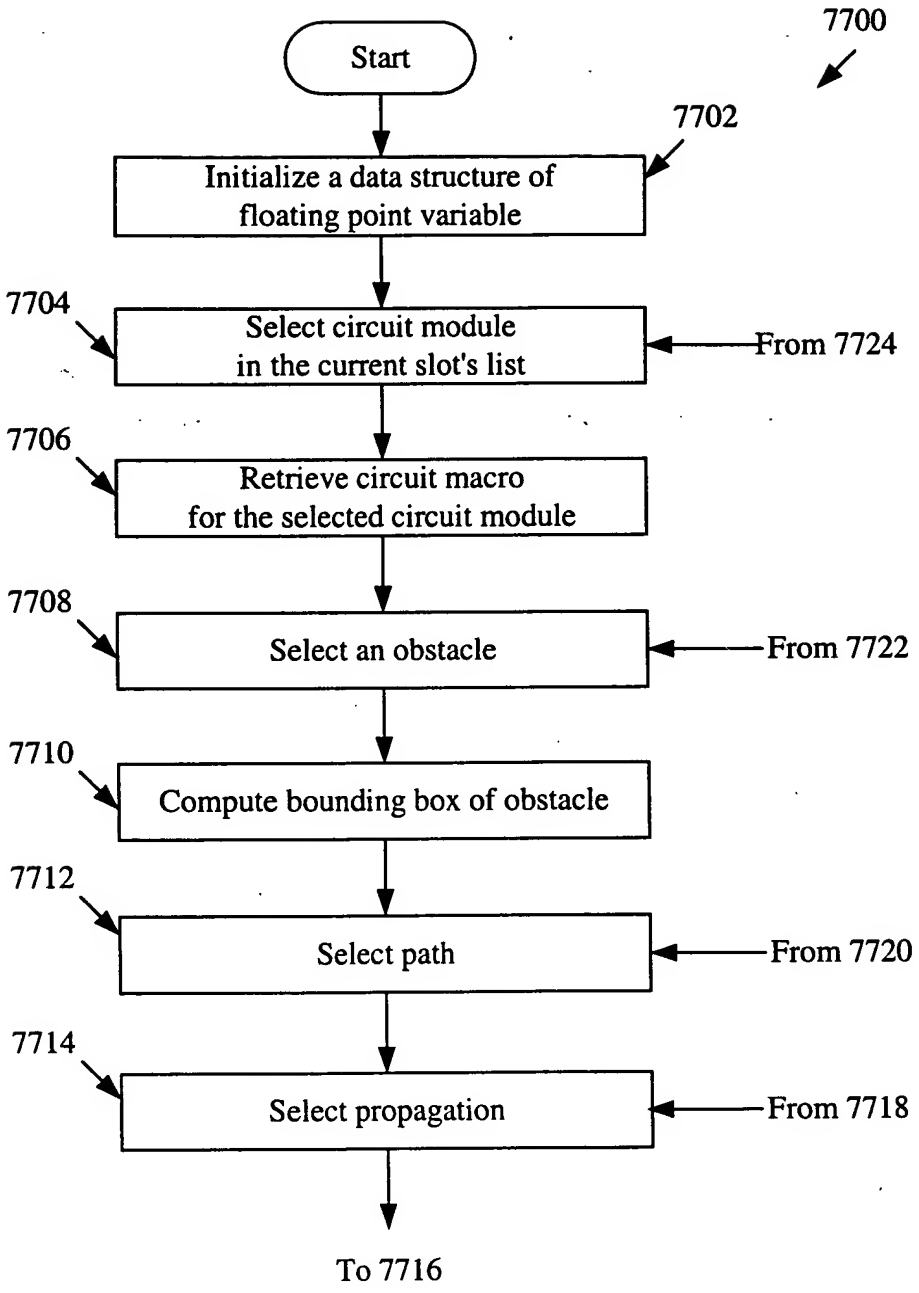

Figure 77A

Figure 77: ***Figure 77A*** / ***Figure 77B***

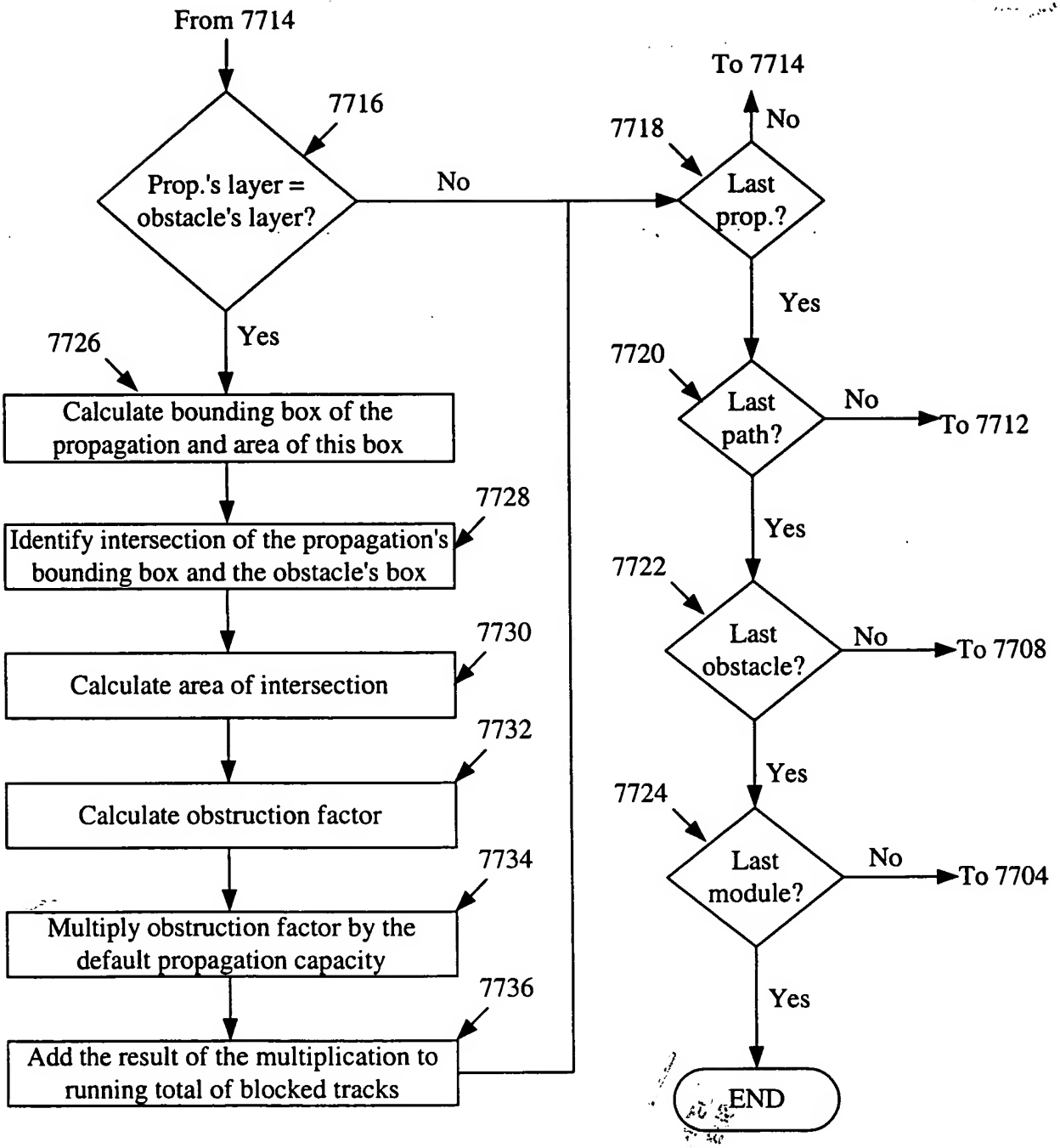
From 7714
7716
Prop.'s layer = obstacle's layer?
No
Yes
7726
Calculate bounding box of the propagation and area of this box
7728
Identify intersection of the propagation's bounding box and the obstacle's box
7730
Calculate area of intersection
7732
Calculate obstruction factor
7734
Multiply obstruction factor by the default propagation capacity
7736
Add the result of the multiplication to running total of blocked tracks
To 7714
No
7718
Last prop.?
Yes
7720
Last path?
No
To 7712
Yes
7722
Last obstacle?
No
To 7708
Yes
7724
Last module?
No
To 7704
Yes
END

Figure 77B

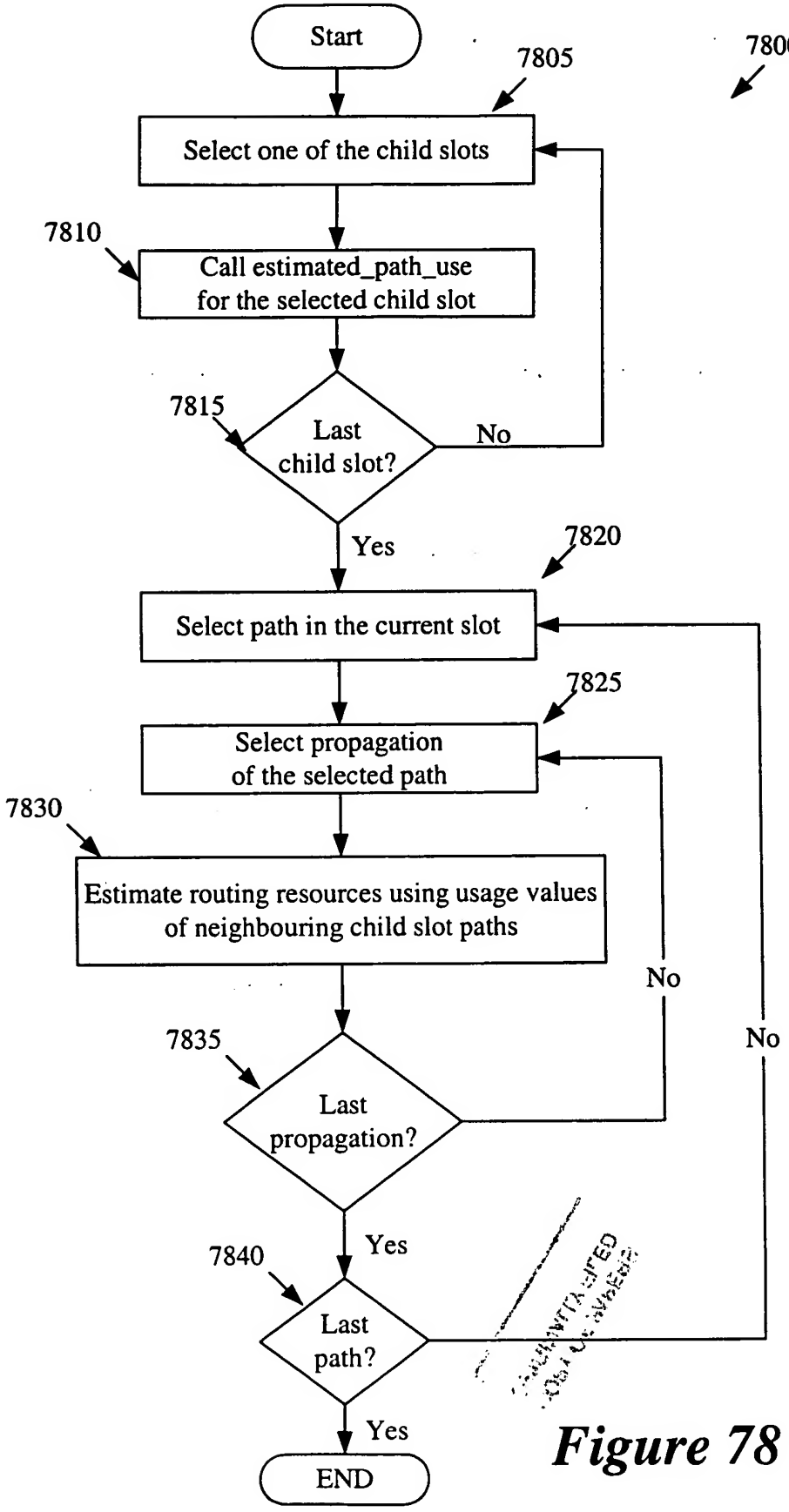
Start
7800
7805
Select one of the child slots
7810
Call estimated_path_use
for the selected child slot
7815
Last
child slot?
No
Yes
7820
Select path in the current slot
7825
Select propagation
of the selected path
7830
Estimate routing resources using usage values
of neighbouring child slot paths
No
No
7835
Last
propagation?
Yes
7840
Last
path?
Yes
END

Figure 78

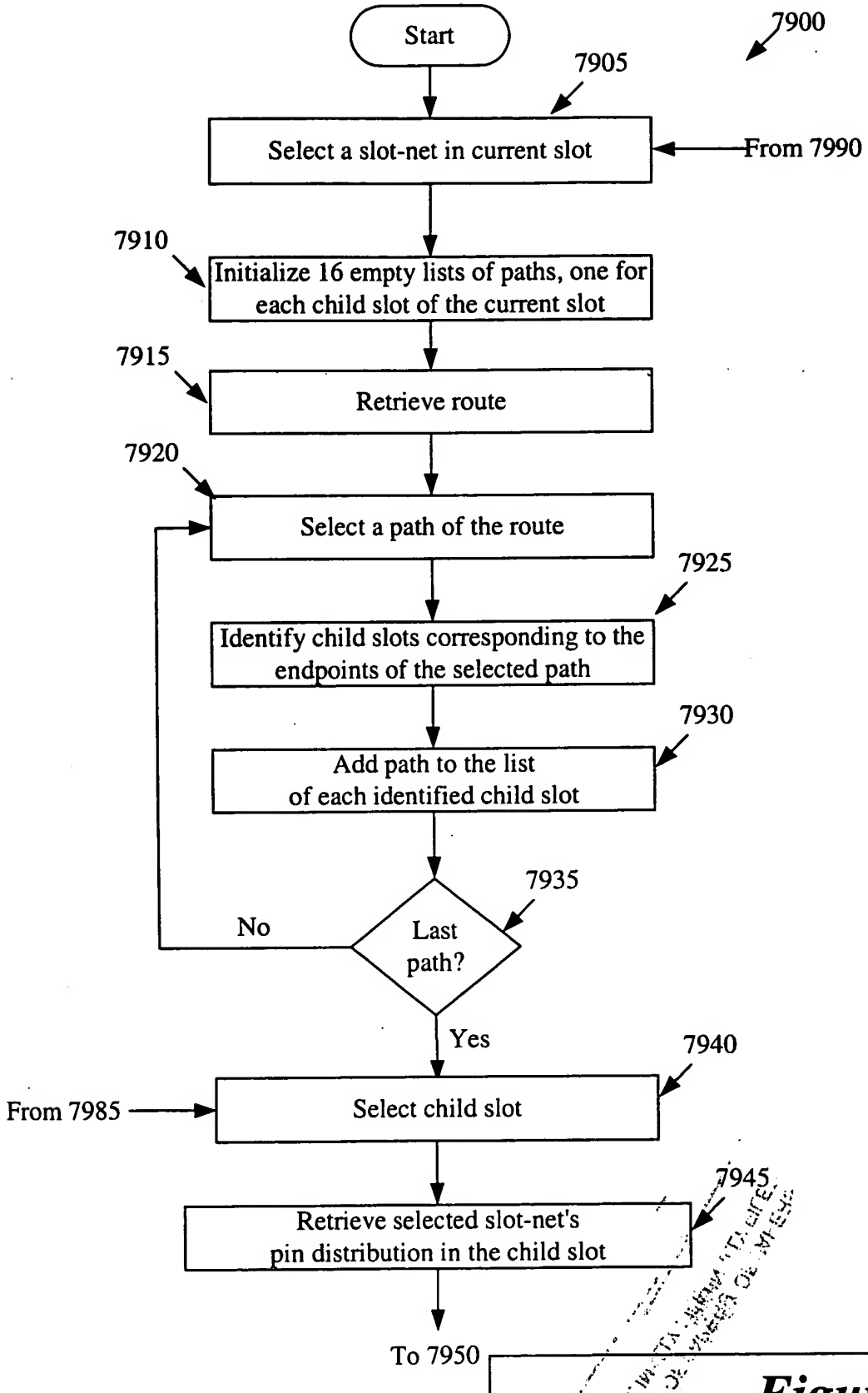

Figure 79A

Figure 79: Figure 79A / Figure 79B

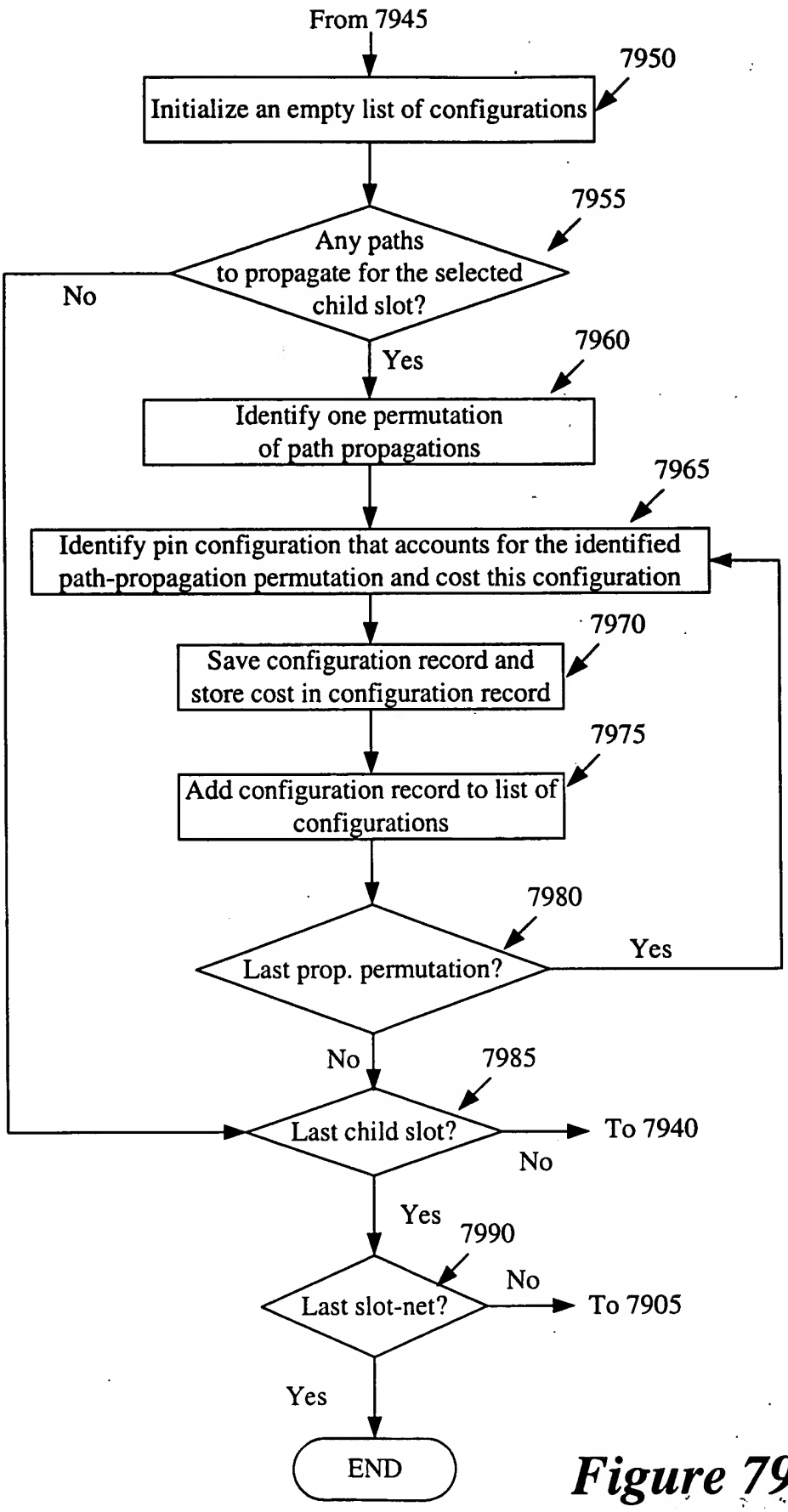
From 7945
7950
Initialize an empty list of configurations
7955
Any paths to propagate for the selected child slot?
No
Yes
7960
Identify one permutation of path propagations
7965
Identify pin configuration that accounts for the identified path-propagation permutation and cost this configuration
7970
Save configuration record and store cost in configuration record
7975
Add configuration record to list of configurations
7980
Last prop. permutation?
Yes
No
7985
Last child slot?
To 7940
No
Yes
7990
Last slot-net?
No
To 7905
Yes
END

Figure 79B

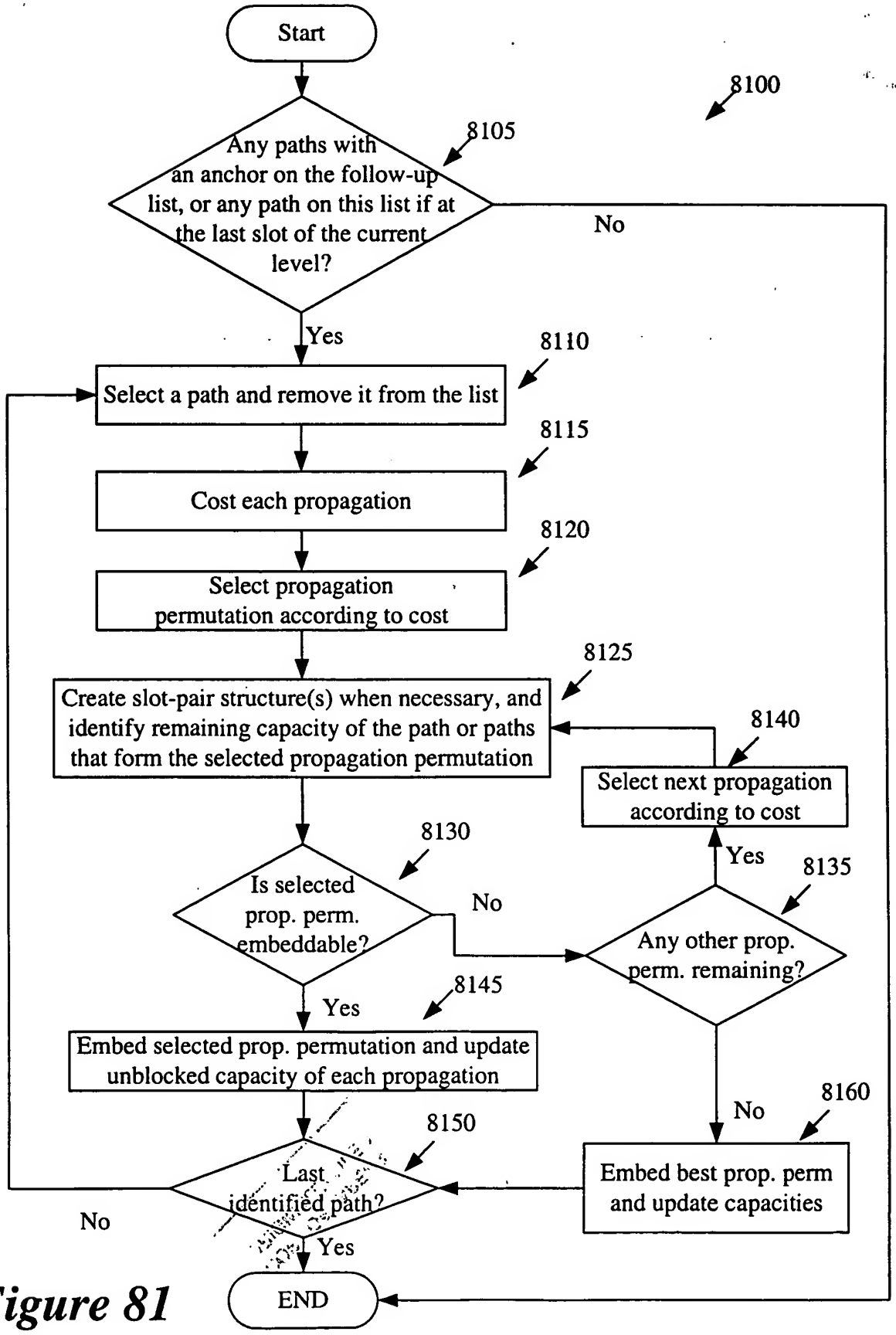
Start
8100
Any paths with an anchor on the follow-up list, or any path on this list if at the last slot of the current level?
8105
No
Yes
8110
Select a path and remove it from the list
8115
Cost each propagation
8120
Select propagation permutation according to cost
8125
Create slot-pair structure(s) when necessary, and identify remaining capacity of the path or paths that form the selected propagation permutation
8140
Select next propagation according to cost
Yes
8130
Is selected prop. perm. embeddable?
No
8135
Any other prop. perm. remaining?
Yes
8145
Embed selected prop. permutation and update unblocked capacity of each propagation
No
8160
8150
Last identified path?
Embed best prop. perm and update capacities
No
Yes
END

Figure 81

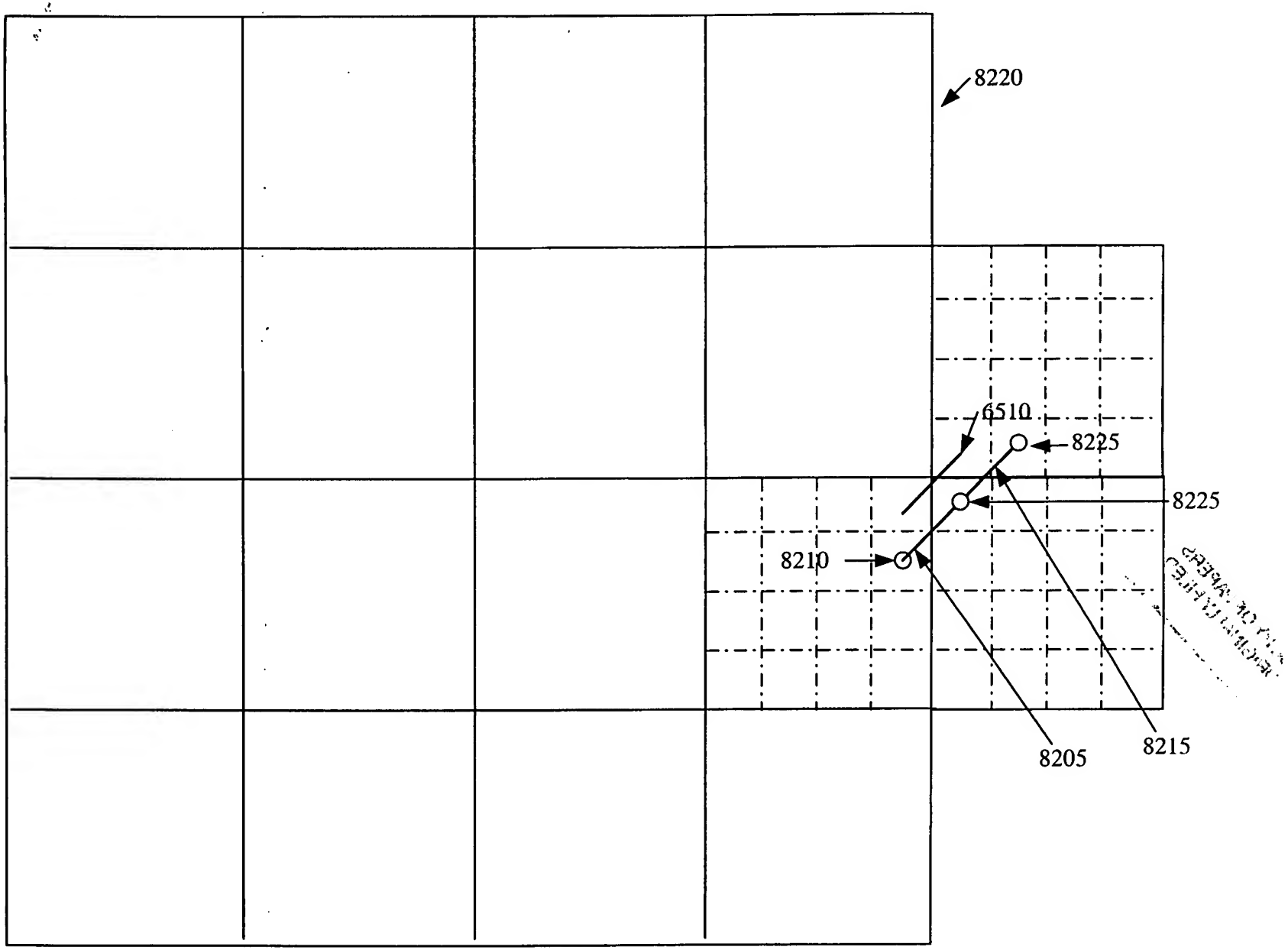

Figure 82

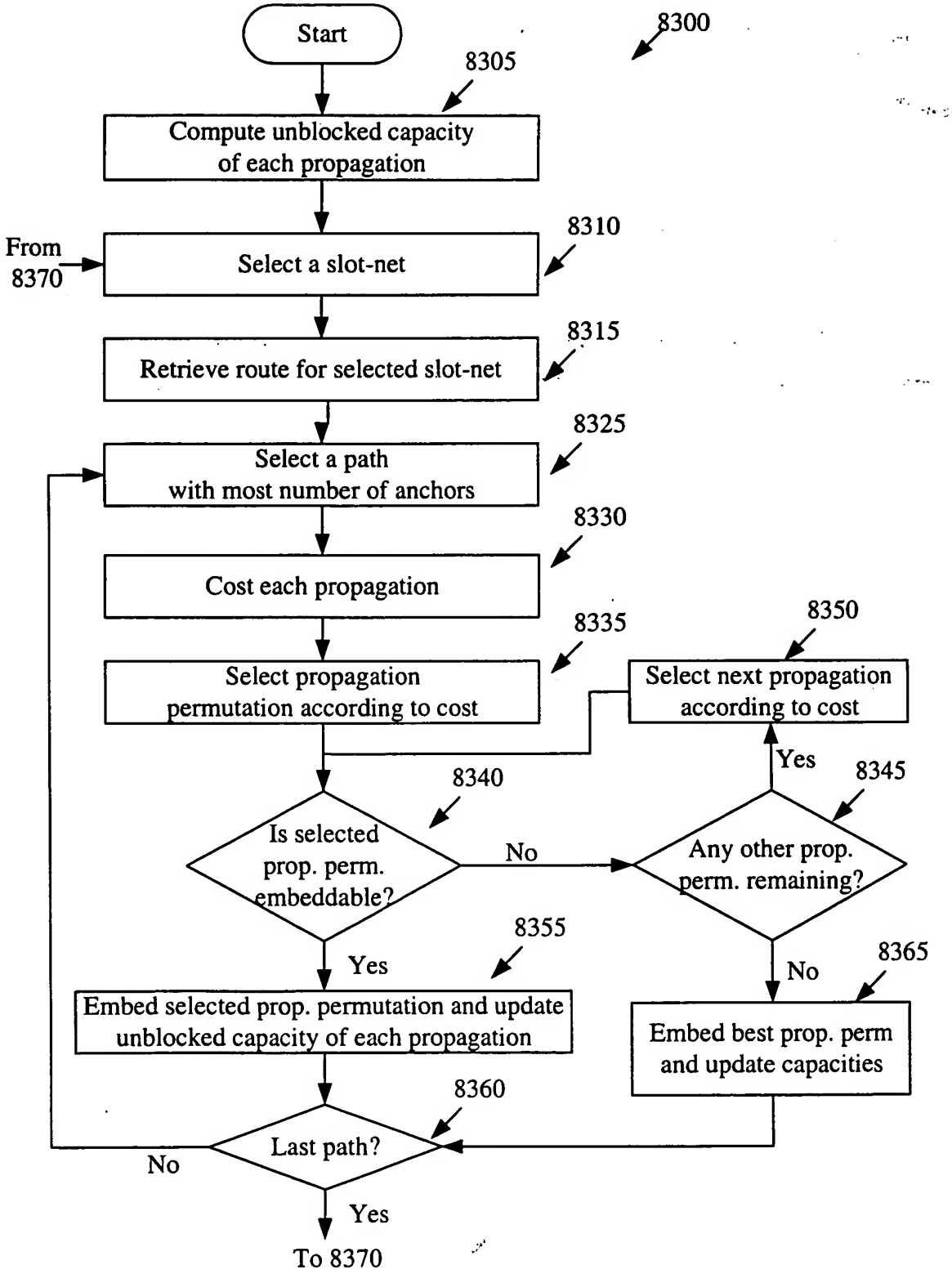

Figure 83A

Figure 83: Figure 83A / Figure 83B

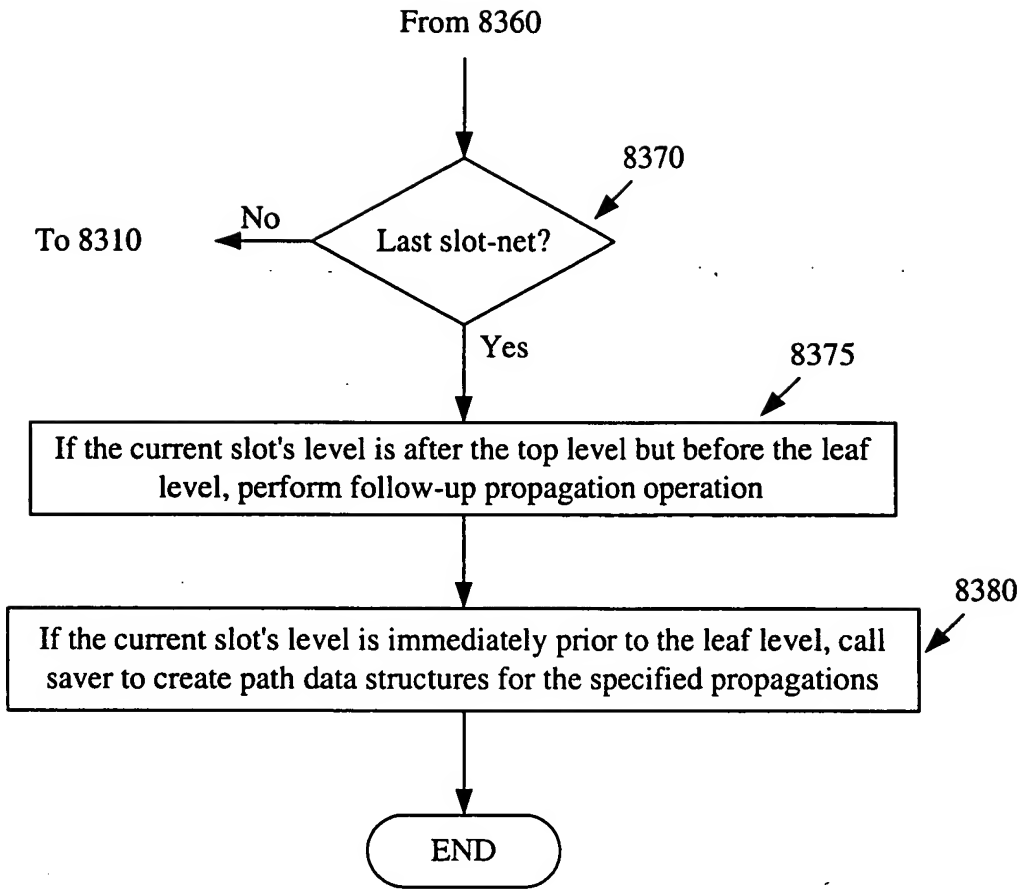
From 8360
8370
No
To 8310
Last slot-net?
Yes
8375
If the current slot's level is after the top level but before the leaf level, perform follow-up propagation operation
8380
If the current slot's level is immediately prior to the leaf level, call saver to create path data structures for the specified propagations
END

Figure 83B

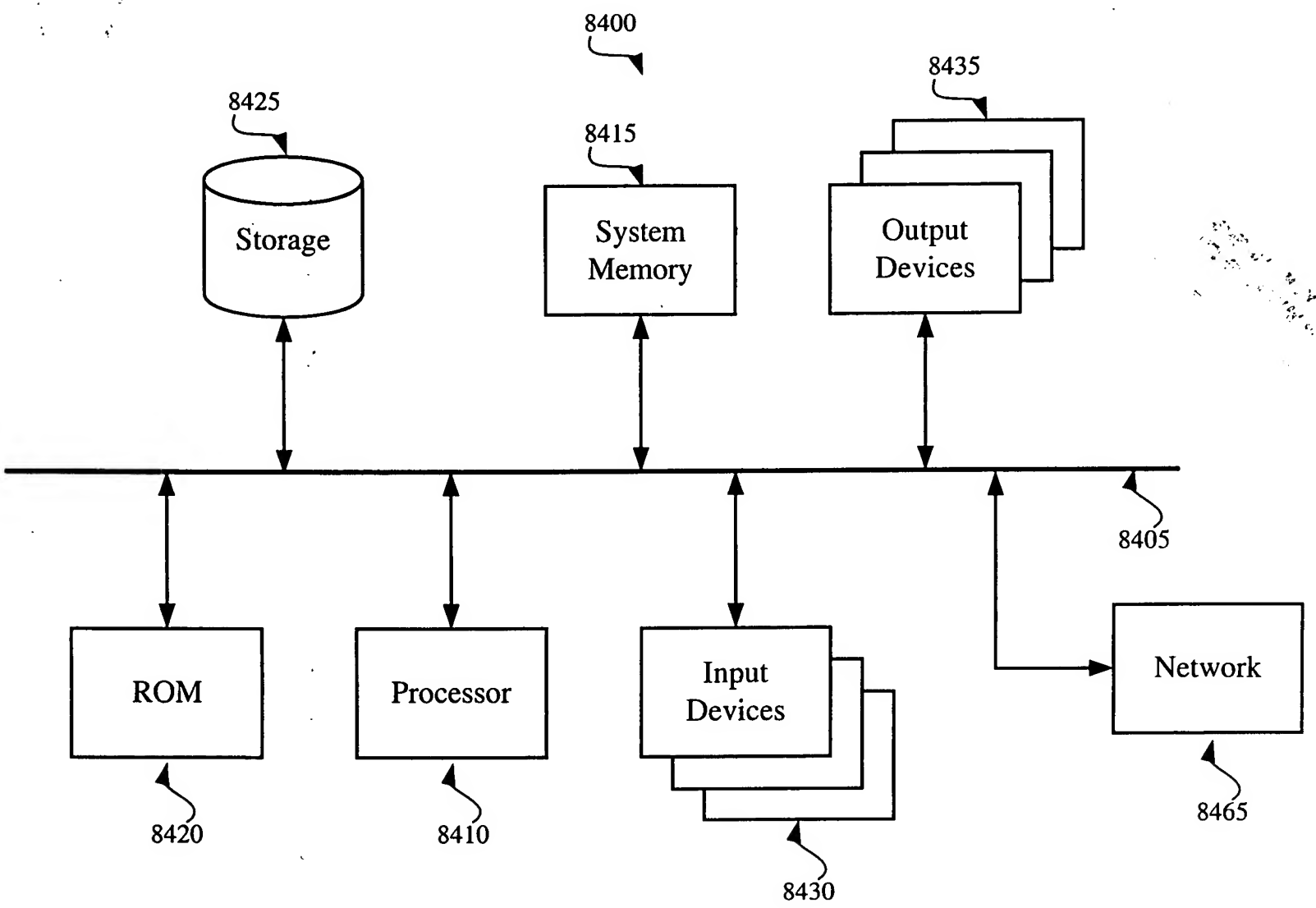
8400
8425
8415
8435
Storage
System Memory
Output Devices
8405
ROM
Processor
Input Devices
Network
8420
8410
8430
8465

Figure 84